Etude d'extensions des langages déterministes

Clément Miklarz

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité Informatique

Préparée au sein de l'Université de Rouen-Normandie

Étude d'extensions des langages déterministes

Présentée et soutenue par
Clément MIKLARZ

Thèse soutenue publiquement le 15 mars 2019 devant le jury composé de		
Mme BÉAL Marie-Pierre	PR à l'Université Paris-Est Marne-la-Vallée	Rapporteure
M. CARON Pascal	PR à l'Université de Rouen-Normandie	Directeur de thèse
Mme De-BOYSSON - FLOURET Marianne	MCF à l'Université Le Havre Normandie	Examinatrice
M. LOMBARDY Sylvain	PR à l'ENSEIRB-MATMECA	Rapporteur
M. MIGNOT Ludovic	MCF à l'Université de Rouen-Normandie	Encadrant de thèse
M. PIN Jean-Éric	DR CNRS à l'IRIF, Université Paris Diderot et CNRS	Examinateur
M. RIGO Michel	PR à l'Université de Liège	Rapporteur

Thèse dirigée par Pascal CARON, laboratoire LITIS

UNIVERSITÉ DE ROUEN NORMANDIE

MIIS

litis

Résumé

Cette thèse a pour but d'étudier des propriétés structurelles d'automates étendant celle du déterminisme, et les langages pouvant être dénotés par une expression rationnelle dont l'automate des positions présente l'une de ces propriétés. Si Book *et al.* ont montré que tous les langages rationnels peuvent être reconnus par un automate des positions non-ambigu, Brüggemann-Klein et Wood ont montré que ceux pouvant l'être par un automate des positions déterministe forment une famille strictement incluse dans celle des rationnels. Nous nous intéressons aux extensions de cette famille, en cherchant à caractériser leurs langages, et à étudier leur hiérarchie interne et leur inclusion entre elles.

Mots-clés : Automates Finis, Langages Rationnels, Expressions Rationnelles, Automate des Positions.

Abstract

This thesis aims to study structural properties of automata extending determinism, and the languages that can be denoted by a regular expression of which the position automaton has one such property. If Book *et al.* showed that all regular languages can be recognized by an unambiguous position automaton, Brüggemann-Klein and Wood showed that only a proper subset of them can be recognized by a deterministic position automaton. We focus on extensions of this subfamily, by seeking to characterize their languages, and to study their internal hierarchy and how they relate to each other.

Keywords : Finite Automata, Regular Languages, Regular Expressions, Position Automaton.

Remerciements

Dans la vie, les hommes sont tributaires les uns des autres. Il y a donc toujours quelqu'un à maudire ou à remercier.
Madeleine Ferron

La reconnaissance est la mémoire du cœur.
Hans Christian Andersen

Par cette première citation, j'essaie de me souvenir que si nos décisions et nos actions font de nous ce que nous sommes, il ne faut pas oublier ce que les autres ont fait pour nous. On dit de quelqu'un qui remercie qu'il exprime sa reconnaissance. Remercier quelqu'un sincèrement, c'est bien reconnaître l'apport de cette personne dans ce que nous avons entrepris.

Par la seconde, je tiens à préciser que les années ont passé depuis que cette thèse a commencé et qu'il est possible que j'oublie de mentionner quelques noms. J'espère que ces personnes me pardonneront.

Je remercie tout d'abord Marie-Pierre Béal, Sylvain Lombardy et Michel Rigo d'avoir accepté de rapporter mon manuscrit de thèse. Je remercie également Marianne De-Boysson-Flouret et Jean-Éric Pin d'avoir accepté d'examiner ma soutenance de thèse.

Je tiens bien sûr à remercier Pascal Caron et Ludovic Mignot de m'avoir accordé leur confiance, même après que je sois sorti du cursus universitaire pendant une année en attendant le financement de cette thèse, et même après que la durée usuelle d'une thèse se soit écoulée. Leur patience m'a offert la liberté d'action dont j'avais besoin pour aller jusqu'au bout.

J'ai également pu compter sur les membres du Département d'Informatique, qui m'ont fait un bon accueil, et avec lesquels j'ai pu échanger librement pendant toutes ces années.

Mes pensées vont également à mes collègues de thèse.
En premier, à Vlad Dragoï pour son accueil chaleureux, son calme et sa

générosité. Je n'aurais pu espérer meilleur collègue pour mon arrivée, et pour la majeure partie de ma thèse.
À Pierre Morisse, pour nos nombreux délires et sa grande connaissance de Kaamelott. S'il n'est pas faux de dire que le gras, c'est la vie, son estomac restera un mystère pour moi.
À Manon Bertin, pour sa bonne humeur et sa capacité à supporter le léger murmure continu et les calembours douteux provenant du bureau voisin. Ses nombreux centres d'intérêts ont alimenté nos discussions.
À Amazigh Amrane, pour son énorme contribution au murmure continu du bureau susmentionné, ses talents de magicien et nos échanges à deux balles.

Un grand merci à mes parents pour m'avoir soutenu, même dans les moments difficiles. Je tiens également à exprimer toute ma gratitude envers Bénédicte pour tout le temps qu'elle m'a accordé quand j'en avais le plus besoin. Enfin, je remercie toutes les personnes de l'Aïkido Club de Rouen et du Karaté Club Sottevillais pour leur attention à chaque séance de pratique.

Un enseignement qui n'enseigne pas à se poser des questions est mauvais.
Paul Valéry

C'est par cette citation que je résumerais l'influence qu'ont pu avoir Philippe Andary, Christophe Hancart, Martine Léonard ainsi que Ludovic sur ma vision de l'enseignement. Je les remercie du temps qu'ils m'ont accordé et des conseils qu'ils m'ont prodigués.

Table des matières

Introduction

L'orthographe est la peau des mots, et les mots sont le langage.
Jean Dutourd

Le langage le plus parfait est celui qui exprime le plus de choses dans le moindre espace.
Antoine Claude Gabriel Jobert

Si les langages nous permettent de communiquer, c'est par l'intermédiaire de mots, s'agençant entre eux selon des règles structurelles, pour former des phrases dont l'interprétation dépendra du sens donné à ces structures. Les mots forment ici les éléments lexicaux, les règles structurelles forment la syntaxe et l'interprétation représente la sémantique. Ces langages que nous parlons et écrivons quotidiennement sont qualifiés de naturels, et sont étudiés par la linguistique dont découle la théorie des langages. Si cette dernière a été fondée dans le but de comprendre leurs fondements logiques, elle est à la base de domaines aussi variés que la théorie de la calculabilité, la théorie de la complexité, ou encore la compilation et la vérification de modèles.

Formellement, une lettre est un symbole, un mot est un produit fini non commutatif de lettres et un langage est un ensemble de mots. En informatique théorique, ces lettres peuvent représenter les actions sur un système, les mots des séquences d'actions et les langages les séquences autorisées à se produire. Un langage peut être représenté entre autres façons :

1. par un ensemble de règles de réécritures, formant une grammaire, pouvant engendrer les mots le constituant,
2. par une machine abstraite reconnaissant les mots le constituant.

La formalisation des langages, par l'utilisation de grammaires, a bénéficié de la linguistique à travers les travaux de Chomsky [25], qui établissent une hiérarchie stricte composée de quatre grandes familles de langages imbriquées. Ces familles ont par la suite été chacune associées à un type de

machine abstraite spécifique, dont la plus générale est la machine de Turing [55]. Dans notre cas, nous nous intéressons à la famille des langages rationnels, dont les éléments peuvent être représentés à l'aide d'expressions rationnelles. Ces dernières ont été introduites par Kleene [41], qui a d'ailleurs montré l'équivalence entre cette famille de langages et celle des langages pouvant être reconnus par un automate à nombre d'états finis. Ce faisant, plusieurs méthodes ont été proposées pour construire aussi bien une expression rationnelle à partir d'un automate, qu'un automate à partir d'une expression rationnelle. Concernant ce dernier cas, nous citerons la construction décrite indépendamment par Glushkov [32] et McNaughton et Yamada [45], permettant d'obtenir un automate appelé l'automate des positions de l'expression rationnelle de départ.

Plusieurs travaux ont alors émergé afin d'étudier les familles d'expressions rationnelles dont les automates des positions présentent certaines propriétés. Book *et al.* [5] ont décrit les expressions rationnelles non ambiguës, dénotant chaque mot du langage de manière unique, et ont conclu que tout langage rationnel peut être dénoté par une telle expression. Une autre recherche découle des spécifications du SGML (Standard Generalized Markup Language [40]) dont est issu, entre autres, le format de données XML (eXtensible Markup Language [6]). Tout document au format XML est décrit par un schéma XML, qui est lui spécifié par un langage de schéma. L'un de ces langages de schéma, le DTD (Document Type Definition [42]), est basé sur une grammaire non contextuelle dont la partie droite des règles est composée d'expressions rationnelles restreintes. Ces dernières ont pour propriété que tout mot w du langage dénoté a une unique décomposition dans l'expression, et que chaque position se déduit de la précédente et du prochain symbole à lire dans w. Ce qui implique que leur automate des positions est déterministe. Brüggemann-Klein et Wood [14] se sont alors intéressés aux langages rationnels pouvant être reconnus par un tel automate des positions. Ils ont montré que ces langages, qu'ils qualifient de 1-non ambigus (et que nous qualifierons de déterministes), forment une famille strictement incluse dans celle des langages rationnels. Les auteurs décrivent notamment une procédure pour décider si un langage rationnel est déterministe à partir de son automate déterministe minimal.

Plusieurs extensions au déterminisme des expressions rationnelles ont été proposées :

— une expression rationnelle k-blocs [31] et déterministe, dans laquelle les lettres sont remplacées par des mots non vides (les blocs) de longueur au plus k, est telle qu'en lisant un mot en entrée, il existe au plus un bloc succédant à la position courante qui est un préfixe de

ce mot. L'automate des positions d'une telle expression contient alors des transitions, étiquetées également par des mots de taille au plus k, tel que pour tout couple de transitions distinctes sortant d'un même état, leurs étiquettes ne sont pas préfixes l'une de l'autre.

— une expression rationnelle k-localement prédictible [36] est telle que la lecture des k prochains symboles du mot en entrée induit un unique chemin et détermine donc totalement quelle position choisir parmi celles succédant directement à la position courante.

— une expression rationnelle (k, l)-localement prédictible [18] est telle que la lecture des k prochains symboles du mot en entrée peut induire l'existence de plusieurs chemins, mais ces derniers doivent alors converger vers un successeur commun à une distance au plus de l (inférieure ou égale à k).

Ces trois familles d'expressions rationnelles forment des familles de langages tel qu'un langage est k-blocs déterministe (respectivement k-localement prédictible, (k, l)-localement prédictible) s'il existe une expression rationnelle k-blocs et déterministe (respectivement k-localement prédictible, (k, l)-localement prédictible) le dénotant.

Giammarresi *et al.* [31] ont étudié la famille des langages blocs déterministes (dont les langages sont k-blocs déterministes pour un certain entier k), et affirmé que cette propriété est décidable pour un langage à partir de son automate déterministe minimal en opérant des suppressions successives d'états. Han et Wood [36] se sont ensuite servis de ce résultat pour étudier la hiérarchie des langages de cette famille par rapport à l'indice k de la longueur maximale des blocs. Ils ont alors montré que pour tout indice k, il existe des langages k-blocs déterministes qui ne sont pas $(k-1)$-blocs déterministes. Ils ont également défini la famille des langages localement prédictibles et montré que tout langage k-blocs déterministe est k-localement prédictible.

Notre étude a donc commencé par la vérification de l'affirmation de Giammarresi *et al.*, qui s'est révélée erronée. Cela a donc remis en cause la preuve d'Han et Wood concernant la hiérarchie de la famille des langages blocs déterministes, d'autant plus que nous avons alors pu montrer que la famille utilisée n'avait pas les bonnes propriétés. Cette thèse a donc consisté à redémontrer certains de ces résultats, et à étudier les propriétés fondamentales des automates à blocs déterministes et celles des automates localement prédictibles.

Le Chapitre 1 contient quelques rappels d'algèbre et définit les notions et objets de base de théorie des langages et de théorie des automates que nous manipulerons par la suite.

Le Chapitre 2 présente quelques propriétés des expressions rationnelles et des automates des positions. Nous y rappelons notamment la caractérisation des automates des positions faite par Caron et Ziadi [20], dont les résultats servent à démontrer les résultats de Brüggemann-Klein et Wood.

Le Chapitre 3 rappelle les travaux sur la famille des langages non-ambigus et celle des langages déterministes. Nous y détaillons notamment le BW-test, dont nous montrerons l'utilité pour l'étude des autres familles dans les Chapitres suivants.

Le Chapitre 4 introduit les notions d'automate et d'expression à blocs, et présente la famille des langages blocs déterministes. Dans un premier temps, nous présentons comment le déterminisme est étendu sur ces automates et étudions leurs propriétés, notamment structurelles, et leurs liens avec les automates déterministes (à lettres). Dans un second temps, nous rappelons les anciens résultats de Giammarresi *et al.* et Han et Wood, montrons en quoi ils sont erronés, et tentons de les corriger. Nous avons notamment été en mesure de démontrer la décidabilité du k-blocs déterminisme d'un langage rationnel, sans toutefois être arrivé à démontrer l'existence d'une borne supérieure permettant de redémontrer la décidabilité du blocs déterminisme. Ce résultat partiel permet néanmoins de redémontrer facilement l'existence d'une hiérarchie interne stricte et que cette famille est strictement incluse dans celle des langages rationnels.

Le Chapitre 5 introduit la notion d'automate prédictible et localement prédictible, et présente la famille des langages localement prédictibles. Nous commençons par définir la notion de délégateur d'un automate. Cette structure permet de décider de la transition à sélectionner à chaque étape, et ce de manière déterministe. Un automate est alors prédictible s'il admet un délégateur. La prédictibilité locale consiste alors en une propriété structurelle de l'automate permettant un calcul plus simple du délégateur. Nous présentons ensuite la famille des langages localement prédictibles, et y démontrons l'existence d'une hiérarchie interne stricte grâce à l'étude des propriétés structurelles des automates localement prédictibles sur un alphabet unaire. Nous terminons en étudiant les relations existant entre cette famille et celle des langages blocs déterministes.

Enfin, le Chapitre 6 présente quelques ouvertures possibles de notre thématique.

Chapitre 1

Préliminaires

1.1 Rappels

1.1.1 Algèbre

1.1.1.1 Ensembles

Un *ensemble* E est une collection d'éléments distincts. Le *cardinal* de E est l'entier naturel $|E|$ représentant le nombre d'éléments dans E. L'unique ensemble ne contenant aucun élément est l'*ensemble vide* noté $\emptyset$. Un ensemble F est un *sous-ensemble* de E si tous les éléments de F sont dans E. Nous disons alors que F est *inclus* dans E, noté $F \subset E$, et deux ensembles sont *égaux* s'ils sont inclus l'un dans l'autre. L'*ensemble* $\mathcal{P}(E)$ *des parties de* E est l'ensemble des sous-ensembles de E.

Pour tout élément x et y, le *couple* (x, y) est une séquence ordonnée de x suivi de y. Deux couples (x_1, x_2) et (y_1, y_2) sont égaux si $x_1 = y_1$ et $x_2 = y_2$. Soit un entier naturel n. La notion de couple se généralise à une séquence ordonnée $(x_1, \cdots, x_n)$ appelée *n-uplet*, avec $()$ l'unique 0-uplet.

Soient E et F deux ensembles. Le *produit cartésien* $E \times F$ représente l'ensemble des couples (x, y) tel que x est un élément de E et y un élément de F. Ce produit est alors vide si et seulement si l'un des deux ensembles est vide. En adéquation avec la définition des n-uplet, ce produit est considéré comme *associatif* : $(E \times F) \times G = E \times (F \times G) = E \times F \times G$. Un couple étant ordonné, le produit cartésien n'est pas commutatif : $E \times F \neq F \times E$. Cette égalité n'est alors vraie que si les deux ensembles sont égaux ou que l'un d'eux est vide. Nous utilisons alors la notation E^n avec n un entier naturel pour représenter la puissance n-ième de l'ensemble E, tel que $E^0 = \{()\}$ et $E^{n+1} = E \times E^n$ pour tout n strictement positif.

1.1.1.2 Relations

Nous définissons une *relation* R d'un ensemble E vers un ensemble F comme un sous-ensemble de $E \times F$. Cette relation est dite *binaire* si $E = F$. Si (x, y) est un couple dans R, alors x est dit *en relation avec* y.

Une relation binaire R sur un ensemble E est :

- *réflexive* si tout élément de E est en relation avec lui-même ;
- *symétrique* si pour tout (x, y) dans R, alors (y, x) est dans R ;
- *anti-symétrique* si pour tout (x, y) et (y, x) dans R, alors x et y sont égaux ;
- *transitive* si pour tout (x, y) et (y, z) dans R, alors (x, z) est dans R ;
- une *relation d'équivalence* si elle est réflexive, symétrique et transitive ;
- une *relation d'ordre* (ou un ordre) si elle est réflexive, anti-symétrique et transitive.

Soit $\equiv$ une relation d'équivalence sur un ensemble E. Pour tout élément x de E, la *classe d'équivalence de x modulo* $\equiv$ est l'ensemble $[x]_{\equiv} = \{y \in E \mid y \equiv x\}$. Le *quotient de E par* $\equiv$ est l'ensemble $E/\equiv = \{[x]_{\equiv} \mid x \in E\}$.

Un ordre $\leq$ sur un ensemble E est dit *total* si pour tout élément x et y de E, soit $x \leq y$ soit $y \leq x$. Un ordre qui n'est pas total est dit *partiel*.

1.1.1.3 Fonctions

Une relation f d'un ensemble E vers un ensemble F est une *fonction* si tout élément de E est en relation avec exactement un élément de F. Pour tout couple (x, y) dans f, y est *l'image* de x par f tandis que x est un *antécédent* de y par f, et nous employons la notation $y = f(x)$.

La fonction f est alors dite :

- *injective* si tout élément de F a au plus un antécédent par f ;
- *surjective* si tout élément de F a au moins un antécédent par f ;
- *bijective* si elle est injective et surjective.

Nous définissons les fonctions suivantes à partir de f :

- l'*image directe par f* est une extension de f de $\mathcal{P}(E)$ vers $\mathcal{P}(F)$ tel que $f(E') = \bigcup_{x \in E'} \{f(x)\}$;
- l'*image réciproque par f* est la fonction f^{-1} de $\mathcal{P}(F)$ vers $\mathcal{P}(E)$ tel que $f^{-1}(F') = \{x \in E \mid f(x) \in F'\}$.

Dans le cas où f est bijective, l'image réciproque est alors définie comme une fonction de F vers E.

1.1.2 Langages

Un *alphabet* Σ est un ensemble fini de symboles insécables, aussi appelés *lettres*. Un *mot* $(a_1, a_2, \ldots, a_n)$ est un n-uplet de lettres, pouvant être représenté comme un produit par l'opérateur de concaténation $\cdot$, qui est donc associatif et non commutatif. En l'absence d'ambiguïté, l'écriture de ce produit est simplifiée de $a \cdot b$ en ab. La longueur du mot $w = a_1 a_2 \cdots a_n$ est l'entier $|w| = n$ correspondant au nombre d'occurrences de lettres apparaissant dedans. L'unique mot de longueur nulle est le mot vide ε.

Soit $w = xyz$ un mot sur l'alphabet Σ. Le mot x (respectivement y et z) est un *préfixe* (resp. *facteur* et *suffixe*) de w, et ces éléments sont dits *propres* s'il sont différents de w. Pour tout deux entiers i et j tel que $1 \leq i \leq j \leq |w|$, nous notons $w[i]$ la i-ème lettre de w et $w[i, j]$ le facteur commençant à la i-ème lettre et se terminant à la j-ème. L'ensemble des préfixes (resp. facteurs et suffixes) de w est noté $\mathrm{Pref}(w)$ (resp. $\mathrm{Fact}(w)$ et $\mathrm{Suff}(w)$).

Un *langage* est un ensemble de mots. Un langage L est dit *préfixe* si pour tout mots distincts w_1 et w_2 appartenant à L, w_1 n'est pas un préfixe de w_2.

Les opérations usuelles sur les ensembles tel que l'union ($\cup$), l'intersection ($\cap$), le complémentaire ($\neg$), la différence ($\setminus$) ou la différence symétrique ($\triangle$) sont également définies sur les langages. De plus, les opérations suivantes sont également définies pour tout langages L et L' :

— la *concaténation* $L \cdot L' = \{ww' \mid w \in L \wedge w' \in L'\}$;

— l'*étoile de Kleene* $L^* = \bigcup_{k \in \mathbb{N}} L^k$ avec $L^0 = \{\varepsilon\}$ et $L^{k+1} = L \cdot L^k$.

L'ensemble des mots sur un alphabet Σ est donc Σ^*, et tout langage est un sous-ensemble de Σ^*.

1.2 Langages rationnels

La famille des langages *rationnels* sur un alphabet Σ [25, 38] est le plus petit ensemble de langages contenant l'ensemble vide $\emptyset$, le singleton $\{\varepsilon\}$, tous les singletons de lettre de Σ, et fermé par l'union, la concaténation et l'étoile de Kleene. Il a été montré que cette famille est également fermée par le complémentaire et l'intersection [38].

Tout langage rationnel admet une représentation finie sous la forme d'une expression :

Définition 1.1. Une *expression rationnelle* E sur un alphabet Σ est définie inductivement à partir des symboles de Σ plus $\emptyset$ et ε, et en utilisant les opérateurs binaires $+$ et $\cdot$ et l'opérateur unaire *.

L'ordre de priorité des opérateurs est le suivant : $^* > \cdot > +$, de sorte que nous omettons les parenthèses, ainsi que l'opérateur $\cdot$, lorsqu'il n'y a aucune ambiguïté.

Une expression rationnelle E peut donc être vu comme un arbre, dont nous définissons les caractéristiques structurelles suivantes de cet arbre :

— la *taille* $|E|$ correspond au nombre de nœuds ;

— la *largeur alphabétique* $||E||$ correspond au nombre de symboles alphabétiques.

Le *langage dénoté* par E est l'ensemble $\mathrm{L}(E)$ défini inductivement selon la Table 1.1, avec a une lettre de Σ, et F et G deux expressions rationnelles.

$$\begin{array}{rr} \mathrm{L}(\emptyset) = \emptyset & \mathrm{L}(F+G) = \mathrm{L}(F) \cup \mathrm{L}(G) \\ \mathrm{L}(\varepsilon) = \{\varepsilon\} & \mathrm{L}(F \cdot G) = \mathrm{L}(F) \cdot \mathrm{L}(G) \\ \mathrm{L}(a) = \{a\} & \mathrm{L}(F^*) = \mathrm{L}(F)^* \end{array}$$

TABLE 1.1 – Calcul du langage dénoté par une expression rationnelle

Exemple 1.2. L'expression rationnelle $(a + bc)^*$ dénote le langage $\{\varepsilon, a, aa, bc, aaa, abc, bca, aaaa, \ldots\}$.

Deux expressions rationnelles E et F sont dites *équivalentes*, et notées $E \equiv F$, si elles dénotent le même langage. Nous définissons alors les équivalences triviales suivantes qui pourront être utilisées par la suite pour simplifier les expressions rationnelles : $E + \emptyset \equiv \emptyset + E \equiv E$, $E \cdot \emptyset \equiv \emptyset \cdot E \equiv \emptyset$, $\emptyset^* \equiv \varepsilon^* \equiv \varepsilon$, $E \cdot \varepsilon \equiv \varepsilon \cdot E \equiv E$.

1.3 Automates

Il est également possible de reconnaître certains langages à l'aide d'un objet mathématique avec une structure de graphe étiqueté, appelé automate fini ou simplement automate.

Définition 1.3. Un *automate* A est un 5-uplet $(\Sigma, Q, I, F, \delta)$ avec :

— Σ un alphabet,

— Q un ensemble fini d'états,

— $I \subset Q$ l'ensemble des états initiaux,

— $F \subset Q$ l'ensemble des états finaux,

— $\delta \subset Q \times \Sigma \times Q$ l'ensemble des transitions étiquetées entre états.

Les éléments constituant l'automate A sont implicitement désignés par Σ_A, Q_A, I_A, F_A et δ_A.

Un automate est représenté graphiquement comme sur la Figure 1.1. Les états sont les rectangles arrondis, dont les initiaux sont représentés avec une flèche entrante sans origine et les finaux avec une double paroi concentrique. Une transition (p, a, q) est représentée par une flèche allant de l'état p vers l'état q, surmontée de l'étiquette a : nous disons qu'elle *sort* de p et *arrive* en q, que p est un *prédécesseur* de q et que q un *successeur* de p. Dans le cas où plusieurs transitions distinctes existent entre p et q, celles-ci sont représentées à l'aide d'une unique flèche de p vers q surmontée d'une liste d'étiquettes.

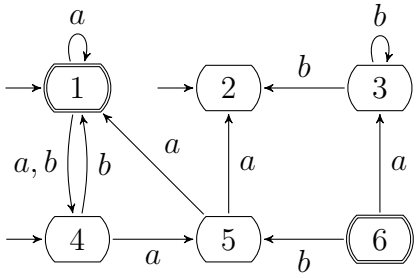


FIGURE 1.1 – Un automate

Nous définissons les relations de comparaisons structurelles suivantes entre deux automates. Nous disons d'un automate B qu'il est :

— *isomorphe* à A s'il existe une bijection φ de Q_A vers Q_B tel que $I_B = \varphi(I_A)$, $F_B = \varphi(F_A)$ et $\delta_B = \{(\varphi(p), a, \varphi(q)) \mid (p, a, q) \in \delta_A\}$.

— un *sous-automate de* A si Q_B, I_B, F_B et δ_B sont des sous-ensembles respectifs de Q_A, I_A, F_A et δ_A.

1.3.1 Chemins et langages d'un automate

La structure de graphe orienté d'un automate permet de définir des parcours entre états dans ce dernier. Un *chemin* d'un état q_0 vers un état q_k est un mot de transitions consécutives $(q_0, a_1, q_1)\ (q_1, a_2, q_2)\ \ldots\ (q_{k-1}, a_k, q_k)$ de δ, dont l'*étiquette* est la concaténation $a_1 a_2 \cdots a_k$ des étiquettes des transitions dans leur ordre de parcours. Le chemin vide, dont l'étiquette est ε, décrit l'ensemble des chemins de n'importe quel état sur lui-même sans emprunter de transition.

Nous définissons alors la *clôture transitive* de δ comme le sous-ensemble δ^t de $Q \times \Sigma^* \times Q$ tel que (p, w, q) appartient à δ^t si et seulement s'il existe un

chemin entre p et q étiqueté par w. Un sous-ensemble δ' de δ^t est équivalent à une fonction de $Q \times \Sigma^*$ vers $\mathcal{P}(Q)$ tel que (p, w, q) est dans δ' si et seulement si q est dans $\delta'(p, w)$. Cette fonction peut ensuite être étendue de $\mathcal{P}(Q) \times \Sigma^*$ vers $\mathcal{P}(Q)$ tel que $\delta'(Q', w) = \bigcup_{q \in Q'} \delta'(q, w)$.

Les chemins dans un automate, en nombre potentiellement infini, définissent les langages associés à chaque état. Un chemin est *acceptant* s'il se termine par un état final, et un chemin acceptant est *réussi* si l'état de départ est initial. Remarquons que le chemin vide est réussi si et seulement s'il existe un état à la fois initial et final. Le *langage droit* d'un état q de A est alors l'ensemble $\mathrm{L}_A(q)$ des mots étiquetant les chemins acceptant partant de q, qui fait partie de l'ensemble $\mathcal{L}_A = \{\mathrm{L}_A(p) \mid p \in Q_A\}$ formant la *famille des langages droits* de A. Le *langage reconnu* par A est, quant à lui, l'ensemble $\mathrm{L}(A)$ des mots étiquetant les chemins réussis de A, et est donc égal à l'union des langages droits des états initiaux.

Deux automates sont alors dits *équivalents* s'ils reconnaissent le même langage, et *$\mathcal{L}$-équivalents* s'ils sont équivalents et ont la même famille de langages droits. Remarquons que selon ces définitions, deux automates isomorphes sont $\mathcal{L}$-équivalents. De plus, un sous-automate de A étant structurellement inclus dans ce dernier, le langage qu'il reconnaît est lui-même inclus dans le langage reconnu par A.

1.3.2 Cycles et orbites

Soit C un sous-ensemble d'états de A non vide. Si pour tout deux états p et q dans C, il existe un chemin allant de p vers q sans passer par un état n'appartenant pas à C, alors C forme un *cycle*. S'il consiste en un unique état sans transition bouclant sur lui-même, alors ce cycle est *trivial*.

Un cycle C est une composante fortement connexe, appelée ici *orbite*, si pour tout état p appartenant à C et q n'appartenant pas à C, q n'est pas accessible depuis p ou p n'est pas accessible depuis q. Cela implique la maximalité d'un tel ensemble d'états, c'est-à-dire que l'intersection de deux orbites distinctes est vide, et un état q appartient donc à une unique orbite notée $\mathrm{O}_A(q)$. Les orbites d'un automate A sont calculables en temps $O(|Q_A| + |\delta_A|)$ par l'algorithme de Tarjan [54].

Les transitions entre états de C sont appelées *transitions internes*. L'ensemble des *transitions de sortie* (respectivement *d'entrée*) de C est noté $\delta_A^{\mathrm{out}}(C) = \{(q, a, p) \in \delta \mid q \in C \wedge p \notin C\}$ (respectivement $\delta_A^{\mathrm{in}}(C) = \{(p, a, q) \in \delta \mid p \notin C \wedge q \in C\}$). Un état est une *porte de sortie* (respectivement *d'entrée*) de C s'il est final ou l'origine d'une transition de sortie (respectivement initial ou la cible d'une transition d'entrée). L'en-

semble des portes de sortie (respectivement d'entrée) de C est noté $\mathrm{Out}(C)$ (respectivement $\mathrm{In}(C)$).

Exemple 1.4. L'automate représenté sur la Figure 1.1 est composé de :

- deux orbites non triviales : $\{1, 4, 5\}$ et $\{3\}$;
- deux orbites triviales : $\{2\}$ et $\{6\}$;
- deux cycles non triviaux distincts des orbites : $\{1\}$ et $\{1, 4\}$;
- deux cycles triviaux distincts des orbites : $\{4\}$ et $\{5\}$.

Remarquons que le sous-ensemble $\{4, 5\}$ n'est pas un cycle car tout chemin allant de 5 vers 4 passe par 1 qui n'appartient pas au sous-ensemble.

1.3.3 Accessibilité et émondage

Le langage reconnu par un automate dépend uniquement des chemins réussis, c'est-à-dire commençant par un état initial et se terminant par un état final. Tout état qui ne fait partie d'aucun de ces chemins est donc inutile à la reconnaissance du langage.

Nous disons qu'un état q est *accessible depuis un état* p s'il existe un chemin allant de p vers q. Et plus spécifiquement, un état est dit *accessible* s'il est accessible depuis un état initial, et *co-accessible* s'il existe un état final accessible depuis cet état.

Un automate est alors *émondé* si tous ses états sont à la fois accessibles et co-accessibles. Si un automate n'est pas émondé, un sous-automate équivalent et émondé est calculable à partir de ce dernier en supprimant tous les états qui sont non accessibles ou non co-accessibles.

Les notions d'accessibilité et de co-accessibilité sont étendues aux cycles par rapport aux états qui les composent. Par définition, si un état d'un cycle est non accessible ou non co-accessible, tous les états de ce cycle le sont nécessairement. Après émondage, les états d'un cycle de l'automate de départ sont donc soit tous présents soit tous absents. La structure interne d'un cycle restant est donc préservée, bien que ses transitions et portes d'entrée et de sortie aient pu changer.

Exemple 1.5. L'automate représenté sur la Figure 1.1 n'est pas émondé : l'état 2 est accessible mais pas co-accessible, l'état 6 est co-accessible mais pas accessible, et l'état 3 n'est ni accessible ni co-accessible. Son équivalent émondé, représenté sur la Figure 1.2, ne contient donc plus que les états 1, 4 et 5 et les transitions internes à ce sous-ensemble d'états. Remarquons que si l'orbite $\{1, 3, 4\}$ a bien conservé sa structure interne, l'état 5 n'en est plus une porte de sortie.

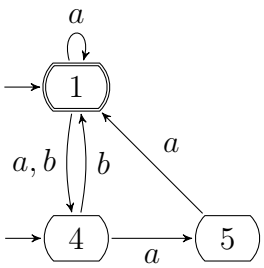


FIGURE 1.2 – Un automate émondé

1.4 Théorème de Kleene et conversions

Les expressions rationnelles dénotent donc des langages avec un système syntaxique, alors que les automates les reconnaissent avec une structure de graphe valué. Le lien entre ces deux formalismes a été étudié par Kleene [41]. En notant $\mathrm{Rat}(\Sigma^*)$ la famille des langages rationnels sur un alphabet Σ et $\mathrm{Rec}(\Sigma^*)$ la famille des langages sur ce même alphabet reconnaissables par un automate, ce lien est résumé par le Théorème suivant :

Théorème 1.6 ([41]). *Soit Σ un alphabet. Alors* $\mathrm{Rat}(\Sigma^*) = \mathrm{Rec}(\Sigma^*)$.

Ainsi, tout langage rationnel peut être reconnu par un automate, et tout automate reconnaît un langage rationnel.

De nombreuses méthodes pour construire un automate reconnaissant le langage dénoté par une expression rationnelle ou l'expression rationnelle dénotant le langage reconnu par un automate ont été proposées. Dans la suite de cette Section, nous étudions l'algorithme proposé par Brzozowski et McCluskey qui est utilisé dans le prochain Chapitre, et la construction de l'automate des positions qui est utilisé dans le reste de cette thèse.

1.4.1 Expression vers automate : l'automate des positions

Cette construction a été présentée indépendamment par McNaughton et Yamada [45] et Glushkov [32], ce dernier lui ayant donné son deuxième nom : l'automate de Glushkov. Celle-ci considère chaque occurrence des symboles alphabétiques dans l'expression comme une position unique, pour l'associer à un état distinct de l'automate à construire. Un même symbole pouvant apparaître plusieurs fois, il est nécessaire de les distinguer.

Une expression rationnelle est dite *linéaire* si chaque symbole alphabétique apparaît au plus une fois. Soit E une expression rationnelle sur un

alphabet Σ. Nous procédons donc d'abord à une indexation des symboles par leurs positions respectives dans l'expression pour obtenir une nouvelle expression rationnelle $E^\sharp$, appelée la *linéarisée* de E, sur un nouvel alphabet Π_E dont les éléments sont les *positions* de $E^\sharp$. Réciproquement, l'opérateur $^\natural$ se définit comme un morphisme de Π_E vers Σ tel que $a_k^\natural = a$, et est ensuite étendu de Π_E^* vers Σ^*.

Linéariser une expression rationnelle ne modifie ni les opérateurs ni la structure. Les expressions E et $E^\sharp$ ont donc des arbres syntaxiques isomorphes, et si F est une sous-expression de E, alors nous définissons l'expression $F^\sharp$ comme l'expression correspondant à la sous-expression de $E^\sharp$ occupant la même position dans l'arbre syntaxique de celle-ci. Cela implique également que si nous appliquons récursivement le morphisme $^\natural$ sur $E^\sharp$, nous retrouvons E. Aussi est-il possible de déduire le langage dénoté par l'expression de départ à partir de celui de sa linéarisée.

Lemme 1.7 ([46]). *Soit E une expression rationnelle. Alors* $\mathrm{L}(E) = (\mathrm{L}(E^\sharp))^\natural$.

Ce Lemme implique donc que l'automate des positions d'une expression rationnelle peut se déduire d'un automate reconnaissant le langage de sa linéarisée. Les fonctions suivantes sont définies pour toute expression rationnelle E sur un alphabet Σ :

— $\mathrm{Null}(E) = \begin{cases} \text{Vrai} & \text{si } \varepsilon \in \mathrm{L}(E) \\ \text{Faux} & \text{sinon} \end{cases}$,

— $\mathrm{First}(E) = \{a \in \Sigma \mid \exists w \in \Sigma^*,\ aw \in \mathrm{L}(E)\}$,

— $\mathrm{Last}(E) = \{a \in \Sigma \mid \exists w \in \Sigma^*,\ wa \in \mathrm{L}(E)\}$,

— $\mathrm{Follow}(E) = \{(a, b) \in \Sigma^2 \mid \exists u, v \in \Sigma^*,\ uabv \in \mathrm{L}(E)\}$.

Appliquées sur une expression rationnelle linéaire, elles permettent d'extraire des propriétés structurelles nécessaires et suffisantes à la description du langage dénoté. Berstel et Pin [4] ont montré que toute expression rationnelle linéaire E sur un alphabet Π dénote un langage dit *local*, tel que $\mathrm{L}(E) \setminus \{\varepsilon\} = \big(\mathrm{First}(E) \cdot \Pi^* \cap \Pi^* \cdot \mathrm{Last}(E)\big) \setminus \big(\Pi^* \cdot (\Pi^2 \setminus \mathrm{Follow}(E)) \cdot \Pi^*\big)$. Ce qui permet d'établir une condition nécessaire et suffisante pour qu'un mot appartienne au langage dénoté par la linéarisée :

Lemme 1.8 ([4]). *Soient E une expression rationnelle linéaire et w un mot non vide de longueur n. Alors w appartient au langage dénoté par E si et seulement si $w[1]$ appartient à* $\mathrm{First}(E)$, *$w[n]$ appartient à* $\mathrm{Last}(E)$, *et pour tout entier i dans $[1, n[$, $(w[i], w[i+1])$ appartient à* $\mathrm{Follow}(E)$.

Remarquons que ce lemme n'est pas nécessairement vrai dans le cas d'une expression rationnelle non linéaire : l'expression rationnelle $E = aa$

admet les ensembles $\mathrm{First}(E) = \{a\}$, $\mathrm{Last}(E) = \{a\}$ et $\mathrm{Follow}(E) = \{(a, a)\}$, mais le mot aaa n'est pas dans le langage dénoté.

Nous définissons alors les automates suivants :

Définition 1.9. Soit E une expression rationnelle sur un alphabet Σ. L'automate $P_E^\sharp = (\Sigma, Q, \{i\}, F, \delta)$ est défini par :

— $Q = \Pi_E \cup \{i\}$,

— $F = \begin{cases} \mathrm{Last}(E^\sharp) \cup \{i\} & \text{si } \mathrm{Null}(E^\sharp) \\ \mathrm{Last}(E^\sharp) & \text{sinon} \end{cases}$,

— $\delta = \{(x, y, y) \mid (x, y) \in \mathrm{Follow}(E^\sharp)\} \cup \{(i, y, y) \mid y \in \mathrm{First}(E^\sharp)\}$.

L'*automate des positions* P_E de E est obtenu en appliquant le morphisme $^\natural$ sur les étiquettes des transitions de $P_E^\sharp$.

À proprement parler, cette Définition décrit un automate des positions de E puisqu'il est possible de linéariser E de différentes façons : il s'agit donc plutôt de l'automate des positions de E à un isomorphisme près.

Proposition 1.10. *Soit E une expression rationnelle. Alors* $\mathrm{L}(P_E^\sharp) = \mathrm{L}(E^\sharp)$ *et* $\mathrm{L}(P_E) = \mathrm{L}(E)$.

Démonstration. Nous montrons d'abord que $\mathrm{L}(P_E^\sharp) = \mathrm{L}(E^\sharp)$. Le mot vide appartient au langage dénoté par $E^\sharp$ si et seulement si $\mathrm{Null}(E^\sharp)$ est vérifiée, ce qui est équivalent au fait que l'état initial i soit final. Soit w un mot de longueur n strictement positive. D'après le Lemme 1.8, w appartient au langage dénoté par $E^\sharp$ si et seulement si $w[1]$ est dans $\mathrm{First}(E^\sharp)$, $w[n]$ est dans $\mathrm{Last}(E^\sharp)$ et pour tout entier i tel que $1 \leq i < n$, $(w[i], w[i+1])$ est dans $\mathrm{Follow}(E^\sharp)$. Ce qui est équivalent à l'existence des transitions $(i, w[1], w[1])$, $(w[1], w[2], w[2])$, …, $(w[n-1], w[n], w[n])$ dans δ, et à la finalité de $w[n]$. Donc w appartient au langage reconnu par $P_E^\sharp$.

Enfin, comme P_E est obtenu en délinéarisant les symboles de toutes les transitions, nous obtenons que $\mathrm{L}(P_E) = (\mathrm{L}(P_E^\sharp))^\natural$. Ce qui, avec le résultat précédent et le Lemme 1.7, permet de conclure que $\mathrm{L}(P_E) = \mathrm{L}(E)$. ■

Exemple 1.11. Soient l'expression $E = (a + b)^* a$ et sa linéarisée $E^\sharp = (a_1 + b_2)^* a_3$. Nous obtenons les résultats suivants :

— $\mathrm{Null}(E^\sharp) = \mathrm{Faux}$,

— $\mathrm{First}(E^\sharp) = \{a_1, b_2, a_3\}$,

— $\mathrm{Last}(E^\sharp) = \{a_3\}$,

— $\mathrm{Follow}(E^\sharp) = \{(a_1, a_1), (a_1, b_2), (a_1, a_3), (b_2, a_1), (b_2, b_2), (b_2, a_3)\}$,

Ce qui nous permet d'obtenir l'automate des positions de E représenté en Figure 1.3.

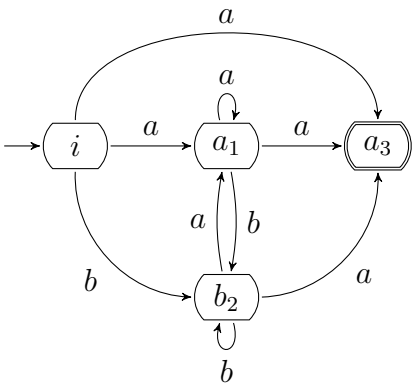


FIGURE 1.3 – L'automate des positions de $(a + b)^*a$

Les propriétés générales des automates des positions et la manière de les calculer sont étudiées au Chapitre 2.

1.4.2 Automate vers expression : l'algorithme de Brzozowski et Mc-Cluskey

La méthode de Brzozowski et Mc-Cluskey [10] consiste à supprimer un à un les états d'un automate pour prolonger les transitions des états restants, étiquetées ici par des expressions rationnelles. Cette méthode s'inspire de celle de McNaughton et Yamada [45], qui calcule exhaustivement les expressions rationnelles dénotant les langages de tout couple d'états. L'utilisation de l'automate comme support permet de n'en calculer qu'un sous-ensemble.

La notion d'automate est donc ici généralisée aux *automates à expressions* dans lesquels les transitions sont étiquetées par des expressions rationnelles. Tout automate est équivalent à un automate à expressions dans lequel les lettres étiquetant les transitions sont interprétées comme des expressions rationnelles. Donc tout langage reconnaissable par un automate l'est aussi par un automate à expressions.

Si la notion de chemin est définie de la même manière que dans les automates finis, nous utilisons pour son étiquette l'associativité à droite du produit de concaténation d'expressions rationnelles. Le langage droit d'un état q est également redéfini comme l'ensemble des mots appartenant au langage dénoté par une expression rationnelle étiquetant un des chemins

acceptant de q. Le langage reconnu par un automate à expressions reste l'union des langages droits de ses états initiaux.

Soient A un automate, et p et q deux états de A. L'expression rationnelle $E^0_{p,q}$ est obtenue en faisant la somme (potentiellement vide) de tous les symboles étiquetant les transition de p vers q dans A. L'algorithme de Brzozowski et Mc-Cluskey commence par créer un automate à expressions A' équivalent à A tel que $A' = (\Sigma_A, Q_A \cup \{i, f\}, \{i\}, \{f\}, \delta \cup \delta_i \cup \delta_f)$ avec $\delta = \{(p, E^0_{p,q}, q) \mid p, q \in Q_A\}$, $\delta_i = (\{i\} \times \{\varepsilon\} \times I_A) \cup (\{i\} \times \{\emptyset\} \times (Q_A \cup \{f\} \setminus I_A))$ et $\delta_f = (F_A \times \{\varepsilon\} \times \{f\}) \cup ((Q_A \cup \{i\} \setminus F_A) \times \{\emptyset\} \times \{f\})$. Ainsi, A' contient un unique état initial, un unique état final et exactement une transition entre chaque état, sauf l'état final qui n'a pas de transition sortante et l'état initial qui n'a pas de transition entrante.

Nous appliquons alors la *règle d'élimination d'état* sur tout état r différent de l'état initial et de l'état final. Cette dernière consiste à créer un nouvel automate à expressions en supprimant r (ainsi que toutes ses transitions entrantes et sortantes) et à mettre à jour les transitions entre tous ses prédécesseurs p et tous ses successeurs s, comme illustrée en Figure 1.4. Les anciennes transitions pouvant être étiquetées par $\emptyset$ ou ε, les nouvelles expressions sont simplifiées à chaque élimination selon les quotients triviaux énoncés à la fin de la Section 1.2 (page 8).

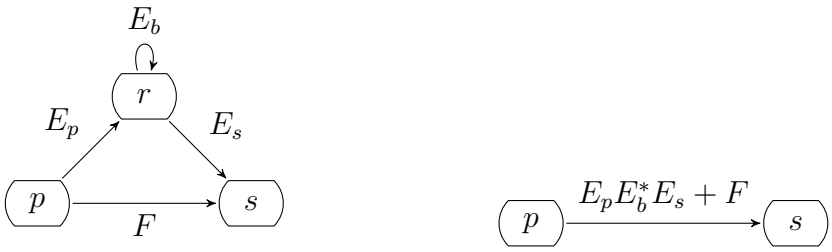


FIGURE 1.4 – Élimination de l'état r

L'élimination d'état préserve l'unicité de la transition entre tout état. De plus, comme il est impossible de supprimer l'état initial ou l'état final, le langage droit des états restants et le langage reconnu sont préservés. L'expression rationnelle dénotant le langage reconnu s'obtient alors en supprimant tous les états de A dans A'.

Nous établissons donc un ordre total $\leq$ sur Q_A définissant l'ordre d'élimination des états. Soit n le nombre d'états de A. Pour tout entier j compris entre 1 et n, nous notons q_j le j-ième état et $Q_j = \{q_m \in Q_A \mid q_m > q_j\}$. Nous obtenons que pour tout entier k compris entre 1 et n, tout état p dans $\{i\} \cup Q_k$ et tout état r dans $Q_k \cup \{f\}$, l'expression rationnelle éti-

quetant la transition entre p et r après élimination des k premiers états est $E^k_{p,r} = E^{k-1}_{p,r} + E^{k-1}_{p,q_k}(E^{k-1}_{q_k,q_k})^* E^{k-1}_{q_k,r}$ (à une simplification près), avec $E^0_{p,r}$ l'expression définie précédemment. L'expression $E^n_{i,f}$ dénote alors le langage reconnu par A' (et donc par A).

Exemple 1.12. Nous souhaitons appliquer l'algorithme sur l'automate A représenté en Figure 1.5. Nous commençons par créer l'automate A', représenté en Figure 1.6 sans les transitions étiquetées par $\emptyset$. Les états sont alors supprimés par ordre croissant, tel que les automates A'_k (représentés en Figure 1.7, 1.8 et 1.9) sont obtenus après avoir supprimé les k premiers états. Ainsi, le langage reconnu par A peut être dénoté par l'expression rationnelle $\varepsilon + (b + (\varepsilon + b)(a + b)) \cdot (ab + ab(a + b))^* \cdot (\varepsilon + a)$.

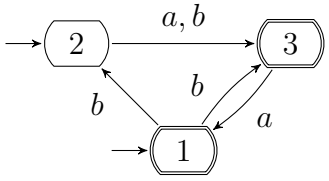


FIGURE 1.5 – L'automate A

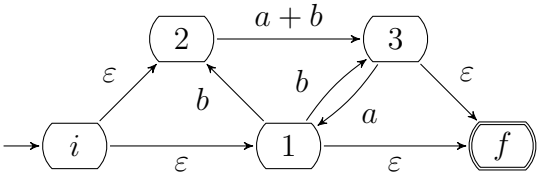


FIGURE 1.6 – L'automate à expressions de départ A'

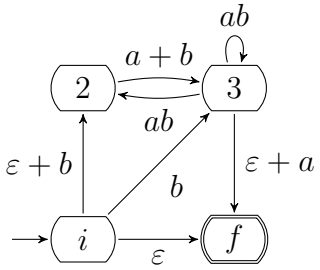


FIGURE 1.7 – L'automate à expressions A'_1

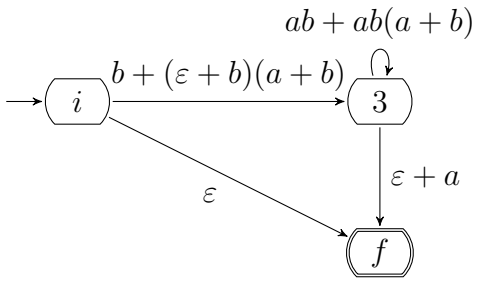


FIGURE 1.8 – L'automate à expressions A'_2

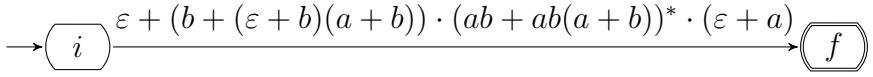


FIGURE 1.9 – L'automate à expressions final A'_3

Remarquons que la largeur alphabétique des expressions rationnelles obtenues peut fortement augmenter après chaque élimination :

Théorème 1.13 ([34])**.** *Soit un automate A avec au moins un état. Le langage reconnu par A peut être dénoté par une expression rationnelle de largeur alphabétique au plus $|\Sigma_A| \times 4^{|Q_A|}$, qui peut être obtenue par l'algorithme de Brzozowski et Mc-Cluskey.*

Si l'ordre d'élimination des états importe peu, il est possible d'établir une heuristique pour pouvoir générer une expression rationnelle plus petite qu'en choisissant aléatoirement. C'est ce que propose Han [35] en identifiant des états particuliers et en tenant compte des longueurs des expressions étiquettant les transitions entrantes et sortantes de chaque état, ou Gruber et Holzer [34] en calculant la profondeur de cycle de chaque état.

1.5 Résiduels d'un langage

Soient L un langage et a une lettre de son alphabet. Le *résiduel* de L par a est l'ensemble $a^{-1}L = \{u \mid au \in L\}$. Cette notion permet de créer une partition du langage qui peut alors se calculer avec les unions disjointes suivantes :

Lemme 1.14. *Soit L un langage. Alors :*

$$L = \begin{cases} \{\varepsilon\} \cup \bigcup_{a\in\Sigma}(\{a\} \cdot a^{-1}L) & \text{si } \varepsilon \in L \\ \bigcup_{a\in\Sigma}(\{a\} \cdot a^{-1}L) & \text{sinon} \end{cases}$$

Dans le cas des opérateurs rationnels, ces résiduels se calculent de la façon suivante :

$$\begin{aligned}
a^{-1}(\emptyset) &= a^{-1}(\{\varepsilon\}) = \emptyset \\
a^{-1}(\{b\}) &= \begin{cases} \{\varepsilon\} & \text{si } b = a \\ \emptyset & \text{sinon} \end{cases} \\
a^{-1}(L_1 \cup L_2) &= a^{-1}(L_1) \cup a^{-1}(L_2) \\
a^{-1}(L_1 \cdot L_2) &= \begin{cases} a^{-1}(L_1) \cdot L_2 \cup a^{-1}(L_2) & \text{si } \varepsilon \in L_1 \\ a^{-1}(L_1) \cdot L_2 & \text{sinon} \end{cases} \\
a^{-1}(L^*) &= a^{-1}(L) \cdot L^*
\end{aligned}$$

TABLE 1.2 – Calcul du résiduel par a d'un langage rationnel

Cette définition s'étend directement au résiduel d'un langage L par un mot avec $\varepsilon^{-1}L = L$ et $(wa)^{-1}L = a^{-1}(w^{-1}L)$ avec a une lettre. L'*ensemble des résiduels* de L est noté $\mathcal{R}_L = \{L' \mid \exists u \in \Sigma^*, L' = u^{-1}L\}$.

De plus, une des propriétés propres à la famille des langages rationnels est que :

Théorème 1.15 ([48, 49])**.** *Un langage est rationnel si et seulement si son ensemble des résiduels est fini.*

Puisque tous les résiduels d'un résiduel d'un langage sont également des résiduels de ce langage, alors ce résiduel a lui-même un nombre fini de résiduels. Cela implique qu'un résiduel d'un langage rationnel est lui-même rationnel, et donc que la famille des langages rationnels est fermée par résiduel.

Il est également possible de trouver un représentant d'un résiduel d'un langage en se servant d'un automate reconnaissant ce langage ou d'une expression rationnelle le dénotant.

1.5.1 Décalé d'un automate

Soient A un automate reconnaissant un langage L, et p un état de A tel qu'il existe une transition (p, a, q). Le calcul du résiduel d'un langage par une lettre consiste à calculer l'ensemble des mots commençant par cette lettre, et à supprimer cette première lettre de chaque mot. Le langage droit de q est donc inclus dans le résiduel du langage droit de p par a, qui est égal à l'union des langages droits des états accessibles depuis p par a. Puisqu'il peut exister plusieurs états accessibles distincts, le langage droit de q n'est donc pas nécessairement un résiduel du langage droit de p, ou même de L.

Les automates dont la famille de langages droits est incluse dans l'ensemble des résiduels du langage reconnu sont qualifiés de *résiduels* [27].

Pour calculer un automate reconnaissant un des résiduels de L à partir de A, il suffit de calculer de nouveaux états initiaux avec la fonction de transition.

Définition 1.16. Soient A un automate et a une lettre de son alphabet. Le *décalé de A par a* est l'automate $(\Sigma_A, Q_A, \delta_A(I_A, a), F_A, \delta_A)$.

Proposition 1.17. *Soient A un automate reconnaissant un langage L, et a une lettre de son alphabet. Le décalé de A par a reconnaît alors le résiduel de L par a.*

1.5.2 Dérivation d'expressions rationnelles

De par la structure des expressions rationnelles, ce calcul se fait de manière inductive et d'une façon similaire au calcul détaillé à la Table 1.2.

Définition 1.18. Soient E une expression rationnelle et a une lettre de son alphabet. La *dérivée de E par a* est l'expression rationnelle $\mathrm{d}_a(E)$ calculée inductivement de la manière suivante :

$$\begin{aligned}
\mathrm{d}_a(\emptyset) &= \mathrm{d}_a(\varepsilon) = \emptyset \\
\mathrm{d}_a(b) &= \begin{cases} \varepsilon & \text{si } b = a \\ \emptyset & \text{sinon} \end{cases} \\
\mathrm{d}_a(F + G) &= \mathrm{d}_a(F) + \mathrm{d}_a(G) \\
\mathrm{d}_a(F \cdot G) &= \begin{cases} \mathrm{d}_a(F) \cdot G + \mathrm{d}_a(G) & \text{si } \mathrm{Null}(F) \\ \mathrm{d}_a(F) \cdot G & \text{sinon} \end{cases} \\
\mathrm{d}_a(F^*) &= \mathrm{d}_a(F) \cdot F^*
\end{aligned}$$

TABLE 1.3 – Calcul de la dérivée par a d'une expression rationnelle

Cette Définition suit donc le modèle du calcul inductif sur les langages rationnels, ce qui implique que :

Proposition 1.19 ([9])**.** *Soient une expression rationnelle E et a une lettre de son alphabet. Alors* $\mathrm{L}(\mathrm{d}_a(E)) = a^{-1}(\mathrm{L}(E))$.

Cela s'étend directement à la dérivation d'une expression par un mot avec $\mathrm{d}_\varepsilon(E) = E$ et $\mathrm{d}_{wa}(E) = \mathrm{d}_a(\mathrm{d}_w(E))$ avec a une lettre de l'alphabet.

De la même manière qu'avec les résiduels de langages, pour toute expression rationnelle, nous pouvons construire une expression rationnelle équivalentes à partir de ses dérivées :

Lemme 1.20. *Soit E une expression rationnelle. Alors E est équivalente à $\varepsilon + \sum_{a \in \mathrm{First}(E)} a \cdot \mathrm{d}_a(E)$ si $\mathrm{Null}(E)$ est vérifiée, sinon à $\sum_{a \in \mathrm{First}(E)} a \cdot \mathrm{d}_a(E)$.*

Pour toute expression rationnelle E sur un alphabet Σ, l'ensemble des dérivées de E, simplifiées selon les quotients triviaux énoncés à la fin de la Section 1.2 (page 8), est noté $\mathcal{D}_E = \{F \mid \exists w \in \Sigma^*, F = \mathrm{d}_w(E)\}$.

Cet ensemble peut contenir un nombre infini d'expressions différentes mais équivalentes.

Exemple 1.21. Soit l'expression rationnelle $E = (a^* + b)^*$. D'après les règles décrites de dérivation décrites plus haut, nous obtenons que :

$$\begin{aligned}
\mathrm{d}_b(E) &= \mathrm{d}_b(a^* + b) \cdot (a^* + b)^* \\
&= (\mathrm{d}_b(a^*) + \mathrm{d}_b(b)) \cdot (a^* + b)^* \\
&= (\mathrm{d}_b(a) \cdot a^* + \varepsilon) \cdot (a^* + b)^* \\
&= (\emptyset + \varepsilon) \cdot (a^* + b)^* \\
&\equiv E
\end{aligned}$$

$$\begin{aligned}
\mathrm{d}_a(E) &= \mathrm{d}_a(a^* + b) \cdot (a^* + b)^* \\
&= (\mathrm{d}_a(a^*) + \mathrm{d}_a(b)) \cdot (a^* + b)^* \\
&= (\mathrm{d}_a(a) \cdot a^* + \emptyset) \cdot (a^* + b)^* \\
&= (\varepsilon \cdot a^* + \emptyset) \cdot (a^* + b)^* \\
&\equiv a^* \cdot E
\end{aligned}$$

$$\begin{aligned}
\mathrm{d}_{aa}(E) &= \mathrm{d}_a(\mathrm{d}_a(E)) \\
&= \mathrm{d}_a(a^* \cdot E) \\
&= \mathrm{d}_a(a) \cdot a^* \cdot E + \mathrm{d}_a(E) \\
&= \varepsilon \cdot a^* \cdot E + \mathrm{d}_a(E \\
&\equiv \mathrm{d}_a(E + a^* \cdot E
\end{aligned}$$

$$\begin{aligned}
\mathrm{d}_{aaa}(E) &= \mathrm{d}_a(\mathrm{d}_{aa}(E)) \\
&= \mathrm{d}_a(\mathrm{d}_a(E) + \mathrm{d}_a(E)) \\
&= \mathrm{d}_a(\mathrm{d}_a(E)) + \mathrm{d}_a(\mathrm{d}_a(E)) \\
&= \mathrm{d}_{aa}(E) + \mathrm{d}_{aa}(E) \\
&= (\mathrm{d}_a(E) + \mathrm{d}_a(E)) + (\mathrm{d}_a(E) + \mathrm{d}_a(E))
\end{aligned}$$

Remarquons que pour tout entier n strictement positif, l'expression $\mathrm{d}_{a^n}(E)$ est une série de sommes contenant 2^{n-1} fois la sous-expression $a^* \cdot E$.

Il a été montré que l'on peut obtenir un ensemble fini de dérivées de E lorsque l'on quotiente $\mathcal{D}(E)$ par quelques relations d'équivalence sur l'opérateur $+$: l'associativité : $(E + F) + G \equiv E + (F + G)$, la commutativité : $E + F \equiv F + E$ et l'idempotence : $E + E \equiv E$.

Théorème 1.22 ([9, 53]). *Le quotient de l'ensemble des dérivées d'une expression rationnelle par les relations d'associativité, de commutativité et d'idempotence de l'opérateur + est fini.*

Remarquons toutefois que le résultat de cet ensemble quotient n'est pas minimal. Dans l'exemple 1.21 notamment, si les expressions $\mathrm{d}_{a^n}(E)$ sont toutes dans la même classe d'équivalence (par les relations d'équivalence sur le +) pour tout entier n strictement positif, l'expression E n'en fait pas partie alors qu'elle est équivalente à $a^* \cdot E$.

1.6 Propriétés des automates

1.6.1 Automates déterministes

Pour décider si un mot appartient au langage reconnu par un automate, il faut déterminer s'il existe un chemin réussi étiqueté par ce mot. Or, tous les états sont susceptibles d'être initiaux et chaque état peut avoir autant de transitions sortantes distinctes et étiquetées par la même lettre qu'il y a d'états dans l'automate. Ce qui se traduit par la complexité suivante :

Théorème 1.23 ([38]). *Soient A un automate et w un mot. L'appartenance de w au langage reconnu par A est décidable en temps $O(|w| \times |Q_A|^2)$.*

La structure d'un automate implique donc qu'il est nécessaire d'explorer l'ensemble des états candidats à chaque étape. Mais la complexité de ce problème peut être simplifiée si cette structure présente comme propriété qu'à chaque étape, le nombre de candidats est au plus d'un :

Définition 1.24. Un automate A est *déterministe* s'il a au plus un état initial et si pour toutes deux transitions distinctes (p, a, q) et (p, b, r), a est différent de b.

Donc pour chaque état et chaque lettre, il y a au plus une transition sortante. Ce qui se traduit sur les chemins de la façon suivante :

Lemme 1.25. *Soient A un AFD, q un état de A et w un mot. Alors $|\delta_A^t(q, w)| \leq 1$.*

Par la suite, un automate (fini) déterministe pourra être désigné par l'abréviation AFD, et l'unique état initial d'un AFD A par i_A. Il s'agit d'une propriété simple à vérifier puisque le nombre de transitions d'un AFD est au plus d'une de chaque lettre de l'alphabet par état.

Théorème 1.26 ([38]). *Le déterminisme d'un automate A est décidable en temps $O(|\Sigma_A| \times |Q_A|)$.*

Puisqu'il n'y a qu'un seul état initial et au plus une transition pour toute lettre, alors le problème de l'appartenance d'un mot au langage reconnu ne dépend plus du nombre d'états ou de transitions de l'automate :

Théorème 1.27 ([38]). *Soient A un AFD et w un mot. L'appartenance de w au langage reconnu par A est décidable en temps $O(|w|)$.*

L'ensemble des langages sur un alphabet Σ reconnaissables par un AFD est noté $\mathrm{DetRec}(\Sigma^*)$. Si les AFD reconnaissent des langages rationnels, il est également établi que tout langage rationnel peut être reconnu par un AFD. Pour tout automate, il est en effet possible de le déterminiser, c'est-à-dire de calculer un AFD équivalent :

Définition 1.28. Soit un automate A. L'automate $(\Sigma_A, \mathcal{P}(Q_A), \{I_A\}, F, \delta)$ tel que $F = \{R \in \mathcal{P}(Q_A) \mid R \cap F_A \neq \emptyset\}$ et $\delta = \{(R, a, S) \mid \delta_A(R, a) = S\}$ est l'*automate des parties* de A.

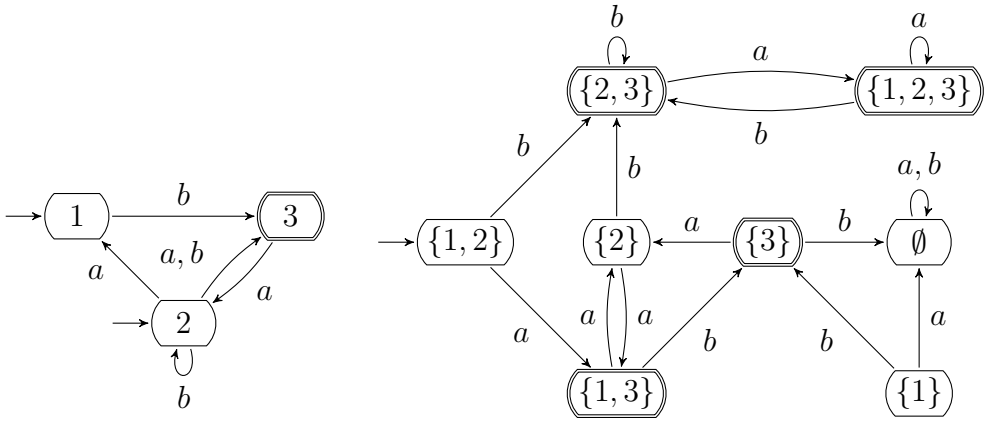


FIGURE 1.10 – Un automate et son automate des parties

Cette construction est exhaustive et tous les états ne sont pas nécessairement accessibles (comme l'état $\{1\}$ en Figure 1.10). De plus, l'état $\emptyset$ n'est jamais co-accessible, et c'est le seul à l'être si tous les états de l'automate de départ sont co-accessibles. Toutefois, le nombre d'états d'un automate des parties émondé peut être exponentiellement plus grand que celui de son automate de départ. Le nombre de ses transitions étant alors borné par le produit du nombre d'états par le nombre de lettres de son alphabet, déterminiser un automate A se fait donc avec une complexité en temps $O(2^{|Q_A|})$.

Proposition 1.29 ([38])**.** *L'automate des parties d'un automate A est déterministe et équivalent à A.*

Théorème 1.30. *Soit Σ un alphabet. Alors* $\mathrm{DetRec}(\Sigma^*) = \mathrm{Rat}(\Sigma^*)$.

1.6.2 Automates déterministes minimaux

S'il existe une transition (p, a, q) dans un automate A, nous pouvons seulement affirmer que le langage droit de q est inclus dans le résiduel du langage droit de p par a. Mais dans le cas d'un AFD, puisqu'il n'existe pas deux transitions distinctes sortant du même état par la même lettre, alors ces deux langages sont égaux. Ce qui amène à la propriété suivante :

Lemme 1.31. *Soit A un AFD émondé. Alors* $\mathcal{L}_A = \mathcal{R}_{\mathrm{L}(A)} \setminus \{\emptyset\}$.

Les AFD font donc partie de la famille des automates résiduels, et tous les AFD émondés reconnaissant le même langage sont $\mathcal{L}$-équivalents. Mais certains ont plus d'états que d'autres car plusieurs états peuvent avoir le même langage droit : ces états sont dits *équivalents*. Un AFD est alors *minimal* s'il n'existe pas d'AFD équivalent avec moins d'états. Ce nombre minimum d'états serait donc au moins égal au cardinal de l'ensemble des résiduels du langage reconnu privé de l'ensemble vide. Nous définissons alors pour tout langage rationnel un automate canonique le reconnaissant :

Définition 1.32. Soit L un langage rationnel sur un alphabet Σ. L'automate $(\Sigma, \mathcal{R}_L \setminus \{\emptyset\}, \{L\}, F, \delta)$ tel que $F = \{L' \mid \varepsilon \in L'\}$ et $\delta = \{(L_1, a, L_2) \mid L_2 = a^{-1}L_1\}$ est l'*automate des résiduels* de L.

Remarquons que le Théorème 1.15 assure que $\mathcal{R}_L \setminus \{\emptyset\}$ est bien fini.

Lemme 1.33. *Soient L un langage rationnel et R son automate des résiduels. Alors R reconnaît L, est déterministe et minimal, et ne contient pas d'états distincts équivalents.*

Démonstration. Un mot w appartient à L si et seulement si ε est dans $w^{-1}L$. Par définition des ensembles δ_R et F_R et du calcul des résiduels d'un langage, un mot w est reconnu par R si et seulement s'il appartient à L.

De plus, puisque le résiduel d'un langage par une lettre est unique, R est déterministe. Ce qui, d'après le Lemme 1.31, permet de déduire que pour tout état L' de R, nous avons $\mathrm{L}_R(L') = L'$. Donc tous les états ont bien des langages droits distincts.

Enfin, le nombre d'états est égal au cardinal de l'ensemble des résiduels du langage reconnu moins l'ensemble vide car l'automate est émondé. Puisque c'est le minimum possible, alors R est minimal. ■

Donc tout langage rationnel est reconnaissable par un automate déterministe minimal ne contenant pas de couple d'états distincts équivalents. Cette propriété se définit dans la suite de la façon suivante :

Définition 1.34. Un automate est *compact* s'il ne contient pas d'états distincts équivalents.

Tous les AFD émondés reconnaissant le même langage étant $\mathcal{L}$-équivalents, si l'un d'eux est compact alors il est minimal. Un AFD est donc minimal si et seulement s'il est émondé et compact. Ce qui permet d'établir que, pour tout langage rationnel L, il existe une bijection entre l'ensemble des états d'un AFD minimal reconnaissant L et $\mathcal{R}_L \setminus \{\emptyset\}$. Et donc que :

Lemme 1.35. *Soit M un AFD minimal. Alors M et l'automate des résiduels de $\mathrm{L}(M)$ sont isomorphes.*

Nous parlons alors de *l'AFD minimal* d'un langage rationnel L, à un isomorphisme près, qui est donc un représentant canonique de ce langage. Par extension, nous parlerons également de l'AFD minimal d'un automate A comme étant celui du langage reconnu par A.

Nous souhaitons maintenant montrer que l'automate des résiduels d'un langage L est calculable à partir de n'importe quel AFD émondé reconnaissant L, en fusionnant les états équivalents. Mais il faut que cette fusion préserve à la fois le langage et le déterminisme. Nous utilisons alors une relation d'équivalence avec des propriétés structurelles supplémentaires :

Définition 1.36. Soit A un automate. Une relation d'équivalence $\equiv$ sur Q_A est une *congruence sur A* si pour tout deux états p et p' dans Q_A tel que $p \equiv p'$, les deux propriétés suivantes sont vérifiées :

1. pour toute transition (p, a, q) dans δ_A, il existe une transition (p', a, q') dans δ_A tel que $q \equiv q'$,
2. si p est final alors p' l'est aussi.

Ces propriétés impliquent qu'une congruence ne peut mettre en relation que des états équivalents :

Lemme 1.37. *Soient A un automate, $\equiv$ une congruence sur A et p et q deux états de A. Si $p \equiv q$, alors $\mathrm{L}_A(p) = \mathrm{L}_A(q)$.*

Démonstration. D'après la condition (2), le mot vide appartient au langage droit de p si et seulement s'il appartient à celui de q. Soit w un mot de longueur n strictement positive dans le langage droit de p. D'après la condition (1), il existe un chemin réussi $(p, w[1], r_1)$, $(r_1, w[2], r_2)$, ...,

$(r_{n-1}, w[n], f)$ dans A si et seulement s'il existe un chemin réussi $(q, w[1], r'_1)$, $(r'_1, w[2], r'_2)$, ..., $(r'_{n-1}, w[n], f')$ dans A tel que $f \equiv f'$ et $r_i \equiv r'_i$ pour tout entier i entre 1 et $n-1$. Donc w appartient au langage droit de p si et seulement s'il appartient à celui de q. ■

Nous pouvons alors définir un automate équivalent basé sur les classes d'équivalence de la congruence employée :

Définition 1.38. Soient A un automate et $\equiv$ une congruence sur A. L'automate $(\Sigma_A, Q_A/\equiv, I_A/\equiv, F_A/\equiv, \delta')$ avec $\delta' = \{([p]_\equiv, a, [q]_\equiv) \mid (p, a, q) \in \delta_A\}$ est l'automate quotient de A par $\equiv$, noté $A/\equiv$.

Lemme 1.39. *Soient A un automate et $\equiv$ une congruence sur A. Alors :*

- *A est équivalent à $A/\equiv$,*
- *si A est déterministe, alors $A/\equiv$ l'est aussi.*

Démonstration. D'après le Lemme 1.37 et la Définition de l'automate quotient, A est équivalent à $A/\equiv$.

Soient $([p]_\equiv, a, [q]_\equiv)$ et $([p]_\equiv, a, [r]_\equiv)$ deux transitions dans $\delta_{A/\equiv}$ avec p, q et r trois états de A. Puisque $\equiv$ est une congruence, il existe deux transitions (p, a, q') et (p, a, r') dans δ_A tel que q' est dans $[q]_\equiv$ et r' est dans $[r]_\equiv$. Comme A est déterministe, alors $q' = r'$ et $[q]_\equiv = [r]_\equiv$. Enfin, puisque A n'a qu'un seul état initial, il en est de même pour $A/\equiv$. ■

Pour tout AFD A, nous définissons alors la relation $\sim_A = \{(p, q) \in Q_A^2 \mid \mathrm{L}_A(p) = \mathrm{L}_A(q)\}$, et nous démontrons qu'elle vérifie les bonnes propriétés :

Lemme 1.40. *Soit A un AFD émondé. La relation $\sim_A$ est une congruence sur A.*

Démonstration. Puisque $\sim_A$ se définit par l'égalité des langages droits, elle est réflexive, symétrique et transitive.

Soient p et p' deux états de A tel que $p \sim_A p'$. Puisqu'ils ont le même langage droit, alors p est final si et seulement si p' l'est aussi. De plus, s'il existe (p, a, r) dans δ_A, alors il existe un mot w tel que aw est dans $\mathrm{L}_A(p)$ car A est émondé. Comme $\mathrm{L}_A(p) = \mathrm{L}_A(p')$ et que A est déterministe, alors il existe une transition (p', a, r') tel que $\mathrm{L}_A(r') = a^{-1}(\mathrm{L}_A(p')) = a^{-1}(\mathrm{L}_A(p)) = \mathrm{L}_A(r)$. Donc $\sim_A$ est bien une congruence sur A. ■

La relation $\sim_A$, également appelée *congruence de Nérode* [49], permet de partitionner l'ensemble des états selon leurs langages droits. D'après les deux lemmes précédents, le quotient d'un AFD émondé par cette congruence est donc un AFD émondé sans états distincts équivalents, c'est-à-dire minimal.

Lemme 1.41. *Soit A un AFD émondé. Alors $A/\sim_A$ est isomorphe à l'AFD minimal de A.*

Si l'AFD de départ contient n états, les classes d'équivalence de ses états sont calculables soit par l'algorithme de Moore [47] en temps $O(n^2)$, soit par l'algorithme de Hopcroft [37] en temps $O(n \times log(n))$. On parle alors de la *minimisation* d'un AFD.

Exemple 1.42. L'automate P en Figure 1.11 est isomorphe à la partie émondée de l'automate des parties de N en Figure 1.10. Il est facile de voir que les états suivants sont équivalents :

— i et 2 car ils ont la même finalité, et leurs transitions sortantes entrent dans les mêmes états pour les mêmes lettres,

— 1 et 4 ont tous les deux un langage droit égal à $\{a, b\}^*$.

Remarquons que le nombre d'états de l'AFD minimal obtenu reste supérieur à celui de l'automate non déterministe N en Figure 1.10.

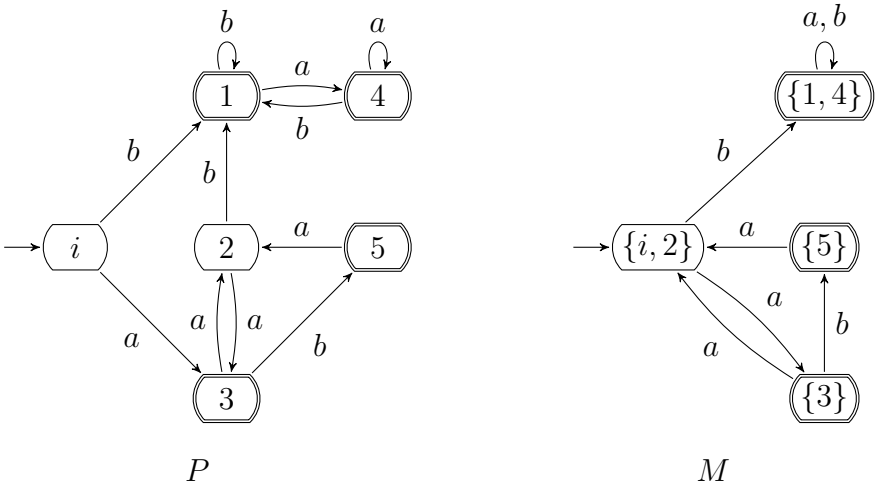


FIGURE 1.11 – Un AFD P et son AFD minimal M

Brzozowski a également décrit une méthode pour obtenir l'AFD minimal d'un langage à partir d'un automate non déterministe [8]. Cette méthode a ensuite été généralisée pour caractériser la famille des automates dont l'automate des parties émondé est l'AFD minimal [12].

1.6.3 Propriétés des automates déterministes minimaux

Puisque tous les AFD émondés reconnaissant le même langage sont $\mathcal{L}$-équivalents, et qu'un AFD minimal est compact, il existe une surjection des états des premiers vers ceux du minimal :

Définition 1.43. Soient A un AFD émondé et M son AFD minimal. La surjection Φ_A associe tout état de A à son équivalent dans M.

Cela implique également qu'un AFD émondé et son minimal partagent des propriétés structurelles sur les chemins, induites par le quotientage du premier par la congruence de Nérode :

Lemme 1.44. *Soient A un AFD émondé et M son AFD minimal. Alors :*

1. *si (p, w, q) est dans δ_A^t, alors $(\Phi_A(p), w, \Phi_A(q))$ est dans δ_M^t*
2. *si (p', w, q') est dans δ_M^t, alors pour tout état p dans $\Phi^{-1}(p')$, il existe un état q dans $\Phi^{-1}(q')$ tel que (p, w, q) est dans δ_A^t.*

Donc si deux états d'un AFD émondé A appartiennent à la même orbite, alors leurs équivalents dans l'AFD minimal M appartiennent également à une même orbite. Donc pour toute orbite K de M, tout antécédent d'un état de K ne peut partager une même orbite de A qu'avec les antécédents d'autres états de K. Comme Φ_A est une surjection, il existe également une surjection associant les orbites de A à celles de M :

Définition 1.45. Soient A un AFD émondé et M son AFD minimal. La surjection Ω_A associe toute orbite O de A à l'orbite K de M tel que $\Phi_A(O)$ est un sous-ensemble de K.

Nous définissons alors une orbite O de A comme étant *maximale* si toute orbite distincte accessible depuis O n'a pas la même image par Ω_A que O. Puisque toute orbite de M a au moins un antécédent dans A par Ω_A, l'un de ces antécédents est nécessairement maximal. Une orbite maximale O doit alors contenir un antécédent par Φ_A de chaque état de $\Omega_A(O)$:

Lemme 1.46. *Soient A un AFD émondé et M son AFD minimal. Alors pour toute orbite maximale O de A, $\Phi_A(O) = \Omega_A(O)$.*

Démonstration. Soient O une orbite maximale de A, K son image par Ω_A et p un état de O. D'après le Lemme 1.44, pour tout $(\Phi_A(p), w, q')$ dans δ_M^t, il existe (p, w, q) dans δ_A^t tel que $\Phi_A(q) = q'$. Puisque O est maximale, si q' est dans K, alors q est dans O. Donc pour tout état de K, il existe un état équivalent dans O et $\Phi_A(O) = \Omega_A(O)$. ■

Chapitre 2

Les automates des positions

Dans ce Chapitre, nous présentons quelques propriétés des automates des positions. Celles-ci sont induites par la structure des expressions rationnelles, et sont nécessaires à l'étude des familles de langages dans les Chapitres suivants. Nous abordons notamment la complexité en temps de la construction, ainsi que les propriétés structurelles caractérisant les automates obtenus.

2.1 Tables de calcul des fonctions

Nous avons défini l'automate des positions d'une expression par rapport aux fonctions Null, First, Last et Follow appliquées sur sa linéarisée. Celles-ci peuvent se calculer inductivement sur la structure de l'expression.

$$\begin{array}{ll} \mathrm{Null}(\emptyset) = \text{Faux} & \mathrm{Null}(F+G) = \mathrm{Null}(F) \vee \mathrm{Null}(G) \\ \mathrm{Null}(\varepsilon) = \text{Vrai} & \mathrm{Null}(F \cdot G) = \mathrm{Null}(F) \wedge \mathrm{Null}(G) \\ \mathrm{Null}(a) = \text{Faux} & \mathrm{Null}(F^*) = \text{Vrai} \end{array}$$

TABLE 2.1 – Calcul de la fonction Null

$$\begin{array}{ll} \mathrm{First}(\emptyset) = \emptyset & \mathrm{First}(F+G) = \mathrm{First}(F) \cup \mathrm{First}(G) \\ \mathrm{First}(\varepsilon) = \emptyset & \mathrm{First}(F \cdot G) = \begin{cases} \mathrm{First}(F) \cup \mathrm{First}(G) & \text{si } \mathrm{Null}(F) \\ \mathrm{First}(F) & \text{sinon} \end{cases} \\ \mathrm{First}(a) = \{a\} & \mathrm{First}(F^*) = \mathrm{First}(F) \end{array}$$

TABLE 2.2 – Calcul de la fonction First

$$\begin{array}{ll} \mathrm{Last}(\emptyset) = \emptyset & \mathrm{Last}(F+G) = \mathrm{Last}(F) \cup \mathrm{Last}(G) \\ \mathrm{Last}(\varepsilon) = \emptyset & \mathrm{Last}(F \cdot G) = \begin{cases} \mathrm{Last}(G) \cup \mathrm{Last}(F) & \text{si } \mathrm{Null}(G) \\ \mathrm{Last}(G) & \text{sinon} \end{cases} \\ \mathrm{Last}(a) = \{a\} & \mathrm{Last}(F^*) = \mathrm{Last}(F) \end{array}$$

TABLE 2.3 – Calcul de la fonction Last

$$\begin{aligned} \mathrm{Follow}(\emptyset) = \mathrm{Follow}(\varepsilon) = \mathrm{Follow}(a) &= \emptyset \\ \mathrm{Follow}(F+G) &= \mathrm{Follow}(F) \cup \mathrm{Follow}(G) \\ \mathrm{Follow}(F \cdot G) &= \mathrm{Follow}(F) \cup \mathrm{Follow}(G) \cup \mathrm{Last}(F) \times \mathrm{First}(G) \\ \mathrm{Follow}(F^*) &= \mathrm{Follow}(F) \cup \mathrm{Last}(F) \times \mathrm{First}(F) \end{aligned}$$

TABLE 2.4 – Calcul de la fonction Follow

Ces Tables se basent sur les Définitions de leurs fonctions respectives, et sont donc valables pour toute expression rationnelle, linéaire ou non. Remarquons toutefois que d'après ces Définitions, les ensembles First, Last et Follow d'une expression dénotant l'ensemble vide devraient être vides. Or, ce n'est pas nécessairement le cas avec ces Tables.

Exemple 2.1. Soit l'expression rationnelle $E = ab \cdot \emptyset \cdot cd$ (qui est donc linéaire). Si le langage dénoté est bien vide, les Tables nous donnent les ensembles $\mathrm{First}(E) = \{a\}$, $\mathrm{Last}(E) = \{d\}$ et $\mathrm{Follow}(E) = \{(a,b),(c,d)\}$. Puisque b n'est suivi par aucune position et ne fait pas partie des dernières positions, et que c ne fait pas partie des premières positions et n'est précédé par aucune position, l'automate obtenu ne serait donc pas émondé mais reconnaîtrait bien le langage vide.

Dans le cas des expressions rationnelles tel que nous les avons définies, c'est-à-dire n'utilisant que les opérateurs $+$, $\cdot$ et *, le symbole $\emptyset$ n'a pour utilité que de permettre de dénoter le langage vide. Cette remarque ne serait toutefois plus valable si nous utilisions d'autres opérateurs tel que le complémentaire ou les multi-tildes-barres de Mignot [46].

Donc, même si le langage reconnu par l'automate des positions ainsi calculé est bien le même que le langage dénoté par l'expression rationnelle de départ, nous restreignons l'utilisation de ce symbole :

Définition 2.2. Une expression rationnelle est *émondée* si elle vaut $\emptyset$ ou ne contient aucune occurrence de ce symbole.

Pour toute expression rationnelle émondée différente de $\emptyset$, son automate des positions est donc également émondé, et les Tables de calcul des fonctions sont cohérentes avec leurs définitions.

Toute expression rationnelle peut être émondée, et nous étudions la relation entre l'automate des positions de l'expression obtenue et celui de l'expression de départ :

Lemme 2.3. *Pour toute expression rationnelle E, il existe une expression rationnelle $E^{\emptyset}$ équivalente et émondée, tel que $P_{E^{\emptyset}}$ est isomorphe à un sous-automate de P_E.*

Démonstration. L'expression $E^{\emptyset}$ se calcule à partir de E en utilisant les équivalences triviales suivantes, dans n'importe quel ordre :

1. $F + \emptyset \equiv \emptyset + F \equiv F$,
2. $F \cdot \emptyset \equiv \emptyset \cdot F \equiv \emptyset$,
3. $\emptyset^* \equiv \varepsilon$.

La preuve se fait alors de manière inductive sur la structure de E. L'application des équivalences (1) et (3) préserve l'automate des positions de départ. Enfin, l'application des équivalences (2) en supprime les états non accessibles ou non co-accessibles. ■

2.2 Forme normale étoilée et complexité de la construction

Une implémentation des fonctions Null, First, Last et Follow selon les Tables de calculs données nécessite dans le pire des cas un temps cubique en la taille de l'expression rationnelle de départ. Mais il est possible d'améliorer cette complexité en observant que le calcul de l'ensemble des Follow d'une expression étoilée est une union non nécessairement disjointe. En effet, dans l'expression $(a_1^*)^*$, le couple (a_1, a_1) appartient à la fois à $\mathrm{Follow}(a_1^*)$ et à $\mathrm{Last}(a_1^*) \times \mathrm{First}(a_1^*)$. Cette redondance entraînant un surcoût dans les calculs, les expressions en étant dépourvues ont été étudiées ainsi que les langages rationnels pouvant être dénotés par de telles expressions.

Soit E une expression rationnelle sur un alphabet Σ. Nous définissons la fonction $\mathrm{Followlast}(E) = \{a \in \Sigma \mid \exists u \in \mathrm{L}(E) \setminus \{\varepsilon\}, \exists v \in \Sigma^*, uav \in \mathrm{L}(E)\}$. Comme son nom l'indique, si on l'applique sur la linéarisée de E, on obtient alors l'ensemble des positions y tel qu'il existe un couple (x, y) dans $\mathrm{Follow}(E^\sharp)$ avec x une position dans $\mathrm{Last}(E^\sharp)$. Cette interprétation n'est toutefois pas transposable aux expressions rationnelles non linéaires :

Exemple 2.4. Soit l'expression rationnelle $E = a(a + b)$ sur l'alphabet Σ. Puisque a et b appartiennent à $\mathrm{Last}(E)$ et (a, b) appartient à $\mathrm{Follow}(E)$, b appartiendrait donc à $\mathrm{Followlast}(E)$ alors que ce dernier est vide.

Brüggemann-Klein [13] définie les expressions dépourvues des redondances citées plus haut de la manière suivante :

Définition 2.5 ([13])**.** Une expression rationnelle est en *forme normale étoilée* si toutes ses sous-expressions de la forme F^* vérifient les propriétés suivantes :

1. $\mathrm{Followlast}(F^\sharp) \cap \mathrm{First}(F^\sharp) = \emptyset$,
2. $\varepsilon \notin \mathrm{L}(F)$.

Cette Définition peut toutefois se simplifier en remarquant que la seconde condition implique la première :

Lemme 2.6. *Une expression rationnelle E est en* forme normale étoilée *si et seulement si pour toute sous-expression F^* de E, ε n'est pas dans* $\mathrm{L}(F)$.

Démonstration. Soit F une expression rationnelle. Nous montrons que si ε n'est pas dans $\mathrm{L}(F)$, alors l'intersection de $\mathrm{Followlast}(F^\sharp)$ avec $\mathrm{First}(F^\sharp)$ est vide. Puisque ε n'est pas dans $\mathrm{L}(F)$, il n'y a que quatre cas à étudier : deux cas de base ($F = \emptyset$ et $F = a$ avec a un symbole alphabétique) qui sont vérifiés trivialement, et deux cas inductifs ($F = F_1 + F_2$ et $F = F_1 \cdot F_2$).

1. Si $F = F_1 + F_2$, alors ε n'est ni dans $\mathrm{L}(F_1)$ ni dans $\mathrm{L}(F_2)$. Donc, par hypothèse d'induction, F_1 et F_2 sont en forme normale étoilée. Comme les expressions $F_1^\sharp$ et $F_2^\sharp$ ne partagent aucune position, l'ensemble $\mathrm{Followlast}(F^\sharp) \cap \mathrm{First}(F^\sharp)$ est donc égal à l'union de $\mathrm{Followlast}(F_1^\sharp) \cap \mathrm{First}(F_1^\sharp)$ et $\mathrm{Followlast}(F_2^\sharp) \cap \mathrm{First}(F_2^\sharp)$. Les expressions F_1 et F_2 étant en forme normale étoilée, cet ensemble est donc vide.
2. Si $F = F_1 \cdot F_2$, alors soit ε n'est pas dans $\mathrm{L}(F_1)$, soit il n'est pas dans $\mathrm{L}(F_2)$. De plus, comme $F_1^\sharp$ et $F_2^\sharp$ ne partagent aucune position, l'ensemble $\mathrm{Follow}(F^\sharp) \cap \mathrm{Last}(F_2^\sharp) \times \mathrm{First}(F_1^\sharp)$ est vide.
   a) Si ε n'est pas dans $\mathrm{L}(F_1)$, alors $\mathrm{First}(F^\sharp) = \mathrm{First}(F_1^\sharp)$ et, par hypothèse d'induction, l'ensemble $\mathrm{Follow}(F^\sharp) \cap \mathrm{Last}(F_1^\sharp) \times \mathrm{First}(F_1^\sharp)$ est vide. Donc l'ensemble $\mathrm{Followlast}(F^\sharp) \cap \mathrm{First}(F^\sharp)$ est vide.
   b) Si ε n'est pas dans $\mathrm{L}(F_2)$, alors $\mathrm{Last}(F^\sharp) = \mathrm{Last}(F_2^\sharp)$ et, par hypothèse d'induction, l'ensemble $\mathrm{Follow}(F^\sharp) \cap \mathrm{Last}(F_2^\sharp) \times \mathrm{First}(F_2^\sharp)$ est vide. Donc l'ensemble $\mathrm{Followlast}(F^\sharp) \cap \mathrm{First}(F^\sharp)$ est vide.

■

Remarquons que ce Lemme est dépendant des opérateurs employés sur les expressions rationnelles. Par exemple, la clôture positive, employée en Section 2.3, invalide ce Lemme.

L'auteure présente alors deux fonctions permettant, à partir d'une expression rationnelle, d'en calculer une nouvelle en forme normale étoilée tout en préservant l'automate des positions.

La fonction $^\circ$ vise à supprimer les liens de succession entre les dernières et les premières positions de l'expression de départ, c'est-à-dire uniquement les liens pouvant être construits par l'application d'une étoile de Kleene. Ainsi, lorsque cette fonction est appliquée à une expression H, elle retourne une expression H' qui n'est pas nécessairement équivalente à H, mais tel que $(H')^*$ et H^* ont le même automate des positions.

$$\emptyset^\circ = \emptyset \quad (F+G)^\circ = F^\circ + G^\circ$$
$$\varepsilon^\circ = \emptyset \quad (F\cdot G)^\circ = \begin{cases} F\cdot G & \text{si } \neg\big(\mathrm{Null}(F)\big) \wedge \neg\big(\mathrm{Null}(G)\big) \\ F^\circ \cdot G & \text{si } \neg\big(\mathrm{Null}(F)\big) \wedge \mathrm{Null}(G) \\ F\cdot G^\circ & \text{si } \mathrm{Null}(F) \wedge \neg\big(\mathrm{Null}(G)\big) \\ F^\circ + G^\circ & \text{si } \mathrm{Null}(F\cdot G) \end{cases}$$
$$a^\circ = a \quad (F^*)^\circ = F^\circ$$

TABLE 2.5 – Calcul de la fonction $^\circ$

La fonction $^\bullet$ s'applique alors récursivement sur les sous-expressions pour parcourir l'ensemble de l'arbre de l'expression de départ, pour ensuite appliquer la fonction $\circ$ aux sous-expressions directement sous une étoile de Kleene. L'expression retournée a donc le même automate des positions que l'expression de départ.

$$\emptyset^\bullet = \emptyset \quad (F+G)^\bullet = F^\bullet + G^\bullet$$
$$\varepsilon^\bullet = \varepsilon \quad (F\cdot G)^\bullet = F^\bullet \cdot G^\bullet$$
$$a^\bullet = a \quad (F^*)^\bullet = (F^{\bullet\circ})^*$$

TABLE 2.6 – Calcul de la fonction $^\bullet$

La complexité de la construction de l'automate des positions d'une expression en forme normale étoilée est alors bornée par la taille de l'arbre de l'expression, qu'il faut parcourir entièrement, et du nombre maximal de transitions à construire qui est proportionnel au carré de la largeur alphabétique de l'expression. Ce qui nous donne :

Théorème 2.7 ([13]). *Soit E une expression rationnelle. L'expression $E^\bullet$ présente les propriétés suivantes :*

1. *$E^\bullet$ est en forme normale étoilée,*
2. *$||E^\bullet|| = ||E||$ et $|E^\bullet| \leq |E|$,*
3. *$P_{E^\bullet} = P_E$,*
4. *$P_{E^\bullet}$ est calculable en temps $O(||E^\bullet||^2 + |E^\bullet|)$.*

Exemple 2.8. Soient l'expression rationnelle $E = F^*$ avec $F = (ab + \varepsilon)^*c^*)$, dont les automates des positions sont représentés respectivement en Figure 2.1 et 2.2. Remarquons que F dénote le mot vide, en plus de ne pas être en forme normale étoilée, et que E n'est donc pas non plus en forme normale étoilée. Si nous appliquons la fonction $^\circ$ sur F, nous obtenons l'expression $ab + c$, dont l'automate des positions est représenté en Figure 2.3. Ce dernier correspond bien à P_F privé de la finalité de son état initial, et des transitions (b_2, a, a_1), (b_2, c, c_3) et (c_3, c, c_3) reliant les états des dernières positions à ceux des premières de F. L'expression $E^\bullet$ vaut donc $(ab + c)^*$.

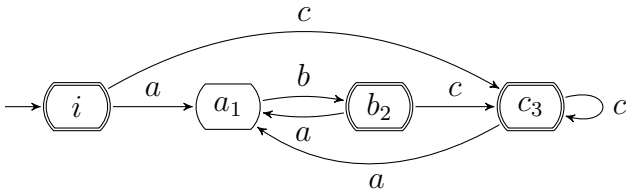


FIGURE 2.1 – L'automate des positions de $((ab + \varepsilon)^*c^*)^*$

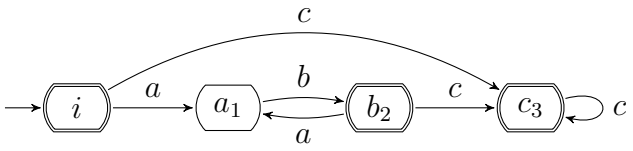


FIGURE 2.2 – L'automate des positions de $(ab + \varepsilon)^*c^*$

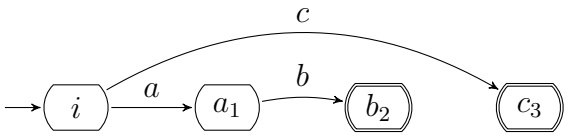


FIGURE 2.3 – L'automate des positions de $ab + c$

Le calcul de la fonction $^\bullet$ semble coûteux à cause de l'application des deux fonctions $^\bullet$ et $^\circ$ dans les sous-expressions étoilées : les sous-expressions étoilées imbriquées seraient parcourues plusieurs fois. L'auteure étudie alors la fonction $^\odot$ tel que $E^\odot = E^{\bullet\circ}$, consistant donc à calculer en une seule étape le résultat de l'exécution séquentielle de $^\bullet$ puis $^\circ$. Il est notamment montré que cette fonction peut se calculer selon la Table 2.7.

$$\emptyset^\odot = \emptyset \qquad (F+G)^\odot = F^\odot + G^\odot$$

$$\varepsilon^\odot = \emptyset \qquad (F\cdot G)^\odot = \begin{cases} F^\bullet \cdot G^\bullet & \text{si } \neg\big(\mathrm{Null}(F)\big) \wedge \neg\big(\mathrm{Null}(G)\big) \\ F^\odot \cdot G^\bullet & \text{si } \neg\big(\mathrm{Null}(F)\big) \wedge \mathrm{Null}(G) \\ F^\bullet \cdot G^\odot & \text{si } \mathrm{Null}(F) \wedge \neg\big(\mathrm{Null}(G)\big) \\ F^\odot + G^\odot & \text{si } \mathrm{Null}(F\cdot G) \end{cases}$$

$$a^\odot = a \qquad (F^*)^\odot = F^\odot$$

TABLE 2.7 – Calcul de la fonction $\odot$

Dans le cas des expressions vérifiant le Lemme 2.6, il est même possible de simplifier le cas de la concaténation [52]. En effet, les dernières positions des sous-expressions ne dénotant pas le mot vide ne sont pas suivies des premières positions. L'application de la fonction $^\circ$ sur une telle sous-expression retournerait donc l'expression de départ. Auquel cas, l'application de la fonction $^\odot$ est équivalente à celle de la fonction $^\bullet$. Ce qui nous donne :

$$(F\cdot G)^\odot = \begin{cases} F^\odot + G^\odot & \text{si } \mathrm{Null}(F\cdot G) \\ F^\bullet \cdot G^\bullet & \text{sinon} \end{cases}$$

Ainsi définies, les fonctions $^\bullet$ et $^\odot$ appliquent au plus une fonction sur chaque sous-expression, impliquant une complexité en temps linéaire. Les propriétés énoncées au Théorème 2.7 permettent de conclure :

Théorème 2.9 ([13])**.** *Soit E une expression rationnelle. Alors :*

1. *$E^\bullet$ est calculable en temps $O(|E|)$,*
2. *P_E est calculable en temps $O(||E||^2 + |E|)$.*

D'autres résultats permettent de prouver cette borne quadratique sans créer une nouvelle expression rationnelle : Chang et Paige [23] font également des unions disjointes par un calcul de l'ensemble des Follow différent lorsqu'il s'agit d'une sous-expression étoilée ; tandis que Champarnaud, Ponty et Ziadi [50, 57] utilisent une structure intermédiaire nommée ZPC.

2.3 Caractérisation

Dans cette partie, nous présentons une interprétation des résultats établis sur la caractérisation de la famille des automates des positions, proposée par Caron et Ziadi [20] puis corrigée par Caron et Flouret [17]. Celle-ci se base sur les propriétés structurelles des automates des positions induites par l'utilisation et le calcul des fonctions Null, First, Last et Follow, tel que si un automate A présente ces caractéristiques, alors il existe une expression rationnelle E déductible de A et dont l'automate des positions est isomorphe à A. Toutefois, cette expression n'est pas canonique car, comme nous avons pu le voir avec la mise en forme normale étoilée, plusieurs expressions rationnelles peuvent avoir le même automate des positions.

La caractérisation qui a été proposée se base sur des expressions rationnelles émondées, et construites à partir des opérateurs classiques ($+$, $\cdot$ et *) et de la clôture positive E^+, équivalente à $E \cdot E^*$. L'utilisation de cet opérateur est compatible avec le Lemme 1.7, et pourrait remplacer complètement celle de l'étoile de Kleene dans une expression rationnelle tout en préservant l'automate des positions obtenu car $E^* = E^+ + \varepsilon$. Nous définissons également les équivalences triviales suivantes : $\emptyset^+ \equiv \emptyset$ et $\varepsilon^+ \equiv \varepsilon$.

Le calcul des fonctions dans le cas de la clôture positive se fait de la manière suivante :

$$\begin{array}{ll} \mathrm{Null}(F^+) = \mathrm{Null}(F) & \mathrm{First}(F^+) = \mathrm{First}(F) \\ \mathrm{Last}(F^+) = \mathrm{Last}(F) & \mathrm{Follow}(F^+) = \mathrm{Follow}(F) \cup \mathrm{Last}(F) \times \mathrm{First}(F) \end{array}$$

TABLE 2.8 – Calculs des fonctions pour la clôture positive

Cette famille d'automates des positions inclut strictement celle des automates des positions sans clôture positive. Nous pourrons néanmoins nous servir de sa caractérisation pour en déduire celle des automates des positions issus d'expressions sans clôture positive, par un test sur la structure de l'expression rationnelle reconstruite.

Nous commençons par présenter quelques propriétés structurelles caractéristiques des automates des positions et induites par leur définition. Un automate est *standard* (respectivement *co-standard*) s'il contient un unique état initial (respectivement final) sans transition entrante (respectivement sortante). Un automate est *homogène* si toutes les transitions arrivant en un même état ont la même étiquette.

Quels que soient les opérateurs utilisés dans les expressions rationnelles, la définition des fonctions Null, First, Last et Follow reste la même et seule

la façon de les calculer diffère. La Définition de l'automate des positions ne dépendant que de ces fonctions, nous concluons que :

Lemme 2.10. *Un automate des positions est homogène et standard.*

La structure d'un automate des positions peut alors être étudiée sans tenir compte des étiquettes des transitions puisqu'elles sont implicitement déduites des états d'arrivée.

2.3.1 Le graphe des positions

Un *graphe orienté* G est un couple (V, U) avec V un ensemble d'états et U un ensemble de transitions inclus dans $V \times V$. Les notions de chemin, d'orbite et de transition d'entrée et de sortie y sont étendues, tandis que les portes d'une orbite sont alors définies uniquement par rapport à leurs transitions sortantes ou entrantes. L'ensemble des prédécesseurs (respectivement successeurs) directs d'un état p de G est noté $V^-(p)$ (respectivement $V^+(p)$). Soit O une orbite de G. L'ensemble des transitions d'entrée (respectivement de sortie) de O est noté $U^{\text{in}}(O)$ (respectivement $U^{\text{out}}(O)$). L'ensemble des prédécesseurs (respectivement successeurs) directs de O est l'ensemble $O^- = \{p \mid \exists (p, q) \in U^{\text{in}}(O)\}$ (respectivement $O^+ = \{q \mid \exists (p, q) \in U^{\text{out}}(O)\}$).

Un automate peut être vu comme un graphe orienté avec un ensemble d'étiquettes sur les transitions et certains états particuliers (initiaux et/ou finaux). Mais alors que les notions d'état initial et final permettent de distinguer les portes des orbites d'un automate, elles ne sont pas conservées dans cette structure. Nous faisons donc en sorte d'ajouter des transitions supplémentaires caractérisant ces propriétés d'initialité et de finalité sur les états de la structure sous-jacente d'un automate.

Un automate des positions étant standard, son unique état initial est donc distinguable. Mais n'étant pas nécessairement co-standard, nous définissons cette structure sous-jacente de la façon suivante :

Définition 2.11. Soit A un automate standard. Le *graphe sous-jacent de* A est le graphe $(Q_A \cup F, U)$ tel que :

— $F = \begin{cases} \{f\} & \text{si } F_A \neq \emptyset \\ \emptyset & \text{sinon} \end{cases}$

— $U = \{(x, y) \mid \exists (x, a, y) \in \delta_A\} \cup (F_A \times F)$.

Ainsi, dans le cas où il existe au moins un état final dans l'automate de départ (et donc que le langage n'est pas vide), tous les anciens états finaux

ont une transition sortante supplémentaire dans le graphe sous-jacent, et uniquement eux, vers une cible commune sans transition sortante. Cela préserve et reflète donc la structure des orbites de l'automate de départ.

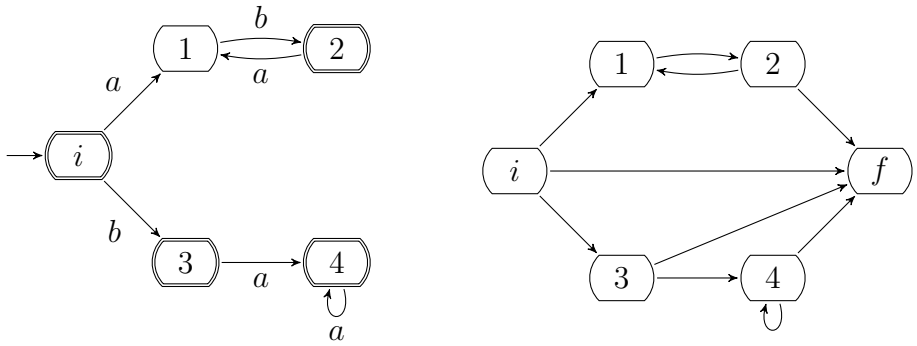


FIGURE 2.4 – Un automate et son graphe sous-jacent

Si un automate est standard et émondé, alors son graphe sous-jacent présente deux états particuliers, non nécessairement distincts : une racine (l'état initial) et une anti-racine (soit le nouvel état ajouté pour co-standardisé, soit l'unique état de l'automate).

Définition 2.12. Le graphe (V, U) est un *hamac* s'il existe un état i et un état f (non nécessairement distincts) dans V tel que les propriétés suivantes sont vérifiées :

1. $V^-(i) = \emptyset$,
2. $V^+(f) = \emptyset$,
3. pour tout état q de V, il existe un chemin de i vers f passant par q.

Lemme 2.13. *Le graphe sous-jacent d'un automate standard et émondé est un hamac.*

Nous appelons *graphe des positions* d'une expression rationnelle le graphe sous-jacent de son automate des positions. Ce qui donne la caractérisation des automates des positions suivante :

Théorème 2.14. *Un automate émondé est un automate des positions si et seulement s'il est standard et homogène, et son graphe sous-jacent est un graphe des positions.*

Il s'agit maintenant de trouver les caractéristiques structurelles des graphes des positions, induites par les différentes sous-expressions possibles d'une expression rationnelle.

2.3.2 Les orbites d'un graphe des positions

Nous commençons par identifier les parties du graphe générées par des sous-expressions de la forme E^* ou E^+ qui sont, d'après les Tables de calcul de leurs fonctions Follow, les seuls opérateurs capables d'engendrer des orbites. Le terme d'*expression de clôture* est utilisé indistinctement pour ces deux formes d'expression rationnelle, et l'on qualifie une sous-expression de clôture de *maximale* si elle n'est pas une sous-expression distincte d'une expression de clôture.

La forme normale étoilée est également étendue en une *forme normale de clôture* tel que toute sous-expression de clôture doit respecter les deux conditions énoncées dans la Définition 2.5. Les formules suivantes complètent alors les Tables de calcul 2.6, 2.5 et 2.7, et permettent d'étendre les Théorèmes 2.7 et 2.9 aux expressions rationnelles avec clôture positive.

$$(E^+)^\circ = E^\circ \quad (E^+)^\odot = E^\odot \quad (E^+)^\bullet = \begin{cases} (E^\odot)^* & \text{si Null}(E) \\ (E^\odot)^+ & \text{sinon} \end{cases}$$

TABLE 2.9 – Calcul des fonctions $^\circ$, $^\odot$ et $^\bullet$ pour la clôture positive

Remarquons que si la clôture positive est utilisée au sein d'une expression rationnelle, le Lemme 2.6 n'est plus valable car les premières positions peuvent alors suivre les dernières sans que le mot vide soit nécessairement dans le langage dénoté. Nous nous intéressons aux liens, identifiés par Brüggemann-Klein et Wood [14] et Caron et Ziadi [20], entre les sous-expressions d'une expression E et les orbites de son graphe des positions :

Lemme 2.15. *Soient E une expression rationnelle, O une orbite de son automate des positions et q un état de O.*

1. *S'il n'existe pas de sous-expression de clôture dans $E^\sharp$ dont q soit une position, alors O est une orbite triviale tel que $O = \{q\}$.*
2. *S'il existe une sous-expression de clôture maximale $H^\sharp$ dans $E^\sharp$ dont q soit une position, alors O est une orbite non triviale tel que $O = \Pi_H$, $\text{In}(O) = \text{First}(H^\sharp)$, $\text{Out}(O) = \text{Last}(H^\sharp)$ et $\text{Follow}(E^\sharp) \cap \Pi_H^2 = \text{Follow}(H^\sharp)$.*

Une orbite non triviale est donc liée à l'existence d'une sous-expression de clôture, dont le calcul de l'ensemble des Follow implique que toutes ses dernières positions sont suivies par toutes ses premières positions. Si ces transitions sont éliminées du graphe des positions d'une expression rationnelle E, le graphe obtenu est également un graphe des transitions associé à une expression calculée à partir de E :

Lemme 2.16. *Soient E une expression rationnelle, G son graphe des positions et O une orbite non triviale de G correspondant à une sous-expression de clôture maximale H de E. Le graphe G', obtenu en supprimant les transitions dans* $\mathrm{Out}(O) \times \mathrm{In}(O)$*, est le graphe des positions d'une expression E' déduite de E en remplaçant H^+ par $H^\bullet$ ou H^* par $(H^\bullet + \varepsilon)$.*

Cette suppression de transitions correspond donc au remplacement d'une sous-expression de clôture maximale par une nouvelle sous-expression, non clôturée, mais pouvant contenir des sous-expressions de clôture. Aussi, définit-on la propriété récursive suivante :

Définition 2.17. Soient $G = (V, U)$ un graphe et O une orbite de G. Alors O est :

- *stable* si $\mathrm{Out}(O) \times \mathrm{In}(O)$ est inclus dans U ;
- *fortement stable* si elle est stable, et si toutes les orbites non triviales obtenues après avoir supprimé les transitions dans $\mathrm{Out}(O) \times \mathrm{In}(O)$ sont également fortement stables.

Remarquons que si une orbite est stable, alors elle est non triviale.

Le calcul de l'ensemble des Follow d'une expression $F \cdot G$ implique également que les dernières positions de F sont suivies par les premières positions de G. De plus, dans le cas d'une sous-expression H d'une expression E, le calcul des premières et dernières positions de E implique que soit $\mathrm{First}(H)$ est inclus dans $\mathrm{First}(E)$ soit $\mathrm{First}(H)$ et $\mathrm{First}(E)$ sont disjoints, et que soit $\mathrm{Last}(H)$ est inclus dans $\mathrm{Last}(E)$ soit $\mathrm{Last}(H)$ et $\mathrm{Last}(E)$ sont disjoints. Les portes de sortie d'une même orbite ont donc la même finalité et les mêmes successeurs directs en dehors de l'orbite, et des portes d'entrée avec les mêmes prédécesseurs directs en dehors de l'orbite.

Ces propriétés structurelles se définissent de la façon suivante :

Définition 2.18. Soient $G = (V, U)$ un graphe et O une orbite de G. Alors O est :

- *transverse en entrée* si $O^- \times \mathrm{In}(O)$ est inclus dans U ;
- *transverse en sortie* si $\mathrm{Out}(O) \times O^+$ est inclus dans U ;
- *fortement transverse* si elle est transverse en entrée et en sortie, et si toutes les orbites non triviales obtenues après avoir supprimé les transitions dans $\mathrm{Out}(O) \times \mathrm{In}(O)$ sont également fortement transverses.

D'après le Lemme 2.15, toute orbite non triviale d'un graphe des positions est donc stable et transverse en entrée et en sortie. De plus, d'après le Lemme 2.16, toute nouvelle orbite non triviale obtenue après la suppression

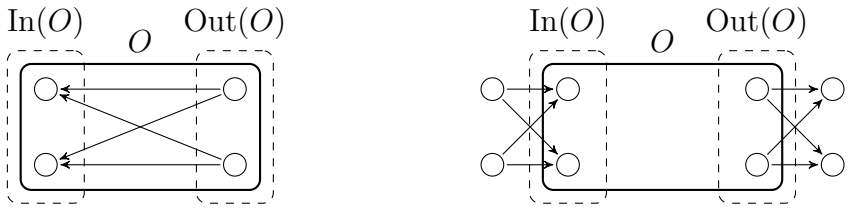


FIGURE 2.5 – Illustrations des propriétés structurelles

de transitions de stabilité est en relation avec une sous-expression de clôture dans une sous-expression de clôture maximale. Ces nouvelles orbites sont donc également stables et transverses en entrée et en sortie, ce qui permet de conclure que :

Lemme 2.19. *Toute orbite non triviale d'un graphe des positions est fortement stable et fortement transverse.*

2.3.3 Graphe acyclique et réduction

Si les orbites permettent d'identifier les opérateurs de clôture, il reste à déterminer les éléments du graphe correspondant aux opérateurs + et $\cdot$. Nous nous intéressons donc aux graphes des positions des expressions rationnelles sans sous-expression de clôture, qui sont donc acycliques, afin de reconstruire l'expression originale à partir des sous-expressions les plus profondes. Celles-ci correspondent à des structures spécifiques du graphe, que l'on réduit ensuite à un unique état.

Nous définissons alors trois règles de réduction, correspondant chacune à une opération particulière :

Définition 2.20. Soit $G = (V, U)$ un graphe acyclique. Alors G est *réductible* s'il peut être réduit à un unique état par une suite d'applications des règles suivantes :

— R_1 : si p et q sont deux états tel que $V^+(p) = \{q\}$ et $V^-(q) = \{p\}$, alors q est supprimé, et p est remplacé par $p \cdot q$ et $U^+(p \cdot q)$ par $U^+(q)$;

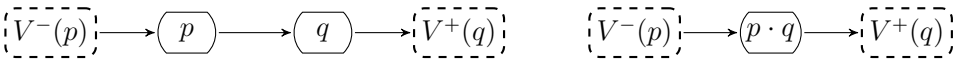


FIGURE 2.6 – La règle R_1

— R_2 : si p et q sont deux états tel que $V^-(p) = V^-(q)$ et $V^+(p) = V^+(q)$, alors q est supprimé et p est remplacé par $p + q$;

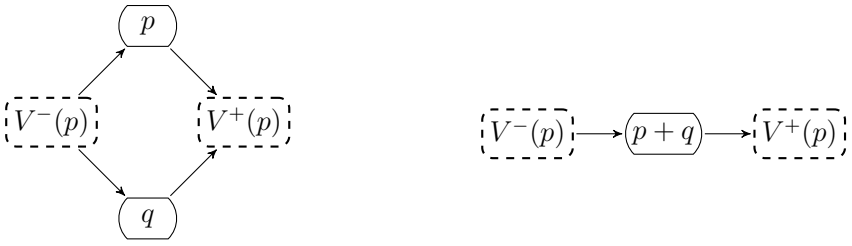


FIGURE 2.7 – La règle R_2

— R_3 : si p est un état tel que pour tout q dans $V^-(p)$, nous avons $V^+(p) \subset V^+(q)$, alors p est remplacé par $p + \varepsilon$ et nous supprimons toute transition (p^-, p^+) dans $V^-(p) \times V^+(p)$ tel qu'il n'existe pas d'état r distinct de p vérifiant les conditions suivantes :

- $r \in V^+(p^-) \cap V^-(p^+)$,
- il n'existe pas de chemin de p vers r, ni de r vers p,
- $|V^-(r)| \times |V^+(r)| \neq 1$.

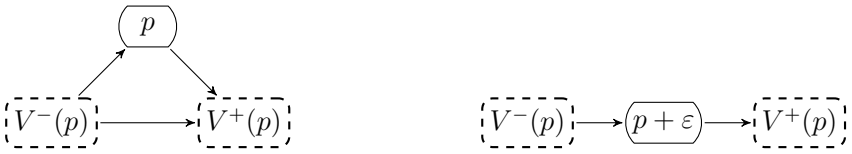


FIGURE 2.8 – La règle R_3

Ces règles préservent et reflètent les caractéristiques d'un graphe des positions acyclique, et les états restants ne représentent alors plus uniquement des positions mais des sous-expressions de l'expression finale. Nous obtenons la caractérisation suivante :

Lemme 2.21. *Un graphe acyclique est un graphe des positions si et seulement s'il est réductible. De plus, les règles de réduction peuvent être appliquées dans n'importe quel ordre.*

Les restrictions sur les transitions à éliminer par la règle R_3 sont dues à la nécessité de conserver ces caractéristiques alors que l'on traite le cas d'une sous-expression E incluse dans une sous-expression $E + F$ tel que ε appartient aux langages dénotés par E et par F. Les transitions entre

les prédécesseurs directs et les successeurs directs de E et de F font donc partie des structures engendrées par ces deux sous-expressions et doivent être conservées jusqu'à la dernière réduction de cette somme.

Exemple 2.22. Soit G le graphe des positions de l'expression $(a+\varepsilon)\ (b+\varepsilon) + (c+\varepsilon)\ (d+\varepsilon)$ représenté en Figure 2.9. Le mot vide est dénoté par les deux sous-expressions de la somme, impliquant donc la transition entre i et f en deux occasions. Cette transition est nécessaire pour appliquer la règle R_3 sur les états a_1 et b_2, comme représenté en Figure 2.10. Elle reste alors nécessaire pour pouvoir appliquer à nouveau la règle R_3 sur c_3 ou sur d_4, après quoi elle sera bien supprimée.

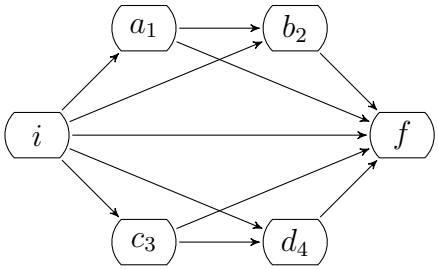


FIGURE 2.9 – Le graphe des positions G de $(a+\varepsilon)(b+\varepsilon)+(c+\varepsilon)(d+\varepsilon)$

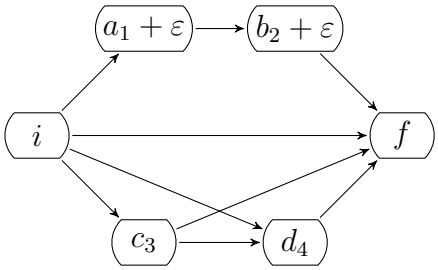


FIGURE 2.10 – La réduction de G par R_3 sur a_1 puis b_2

2.3.4 Caractérisation et algorithme de remontée

D'après le Lemme 2.19, toute orbite non triviale d'un automate des positions est fortement stable. Nous nous intéressons au graphe acyclique obtenu par la suppression récursive des transitions de stabilité :

Définition 2.23. Soit $G = (V, U)$ un graphe dont toutes les orbites non triviales sont fortement stables. Le *graphe sans orbite de* G est obtenu par

élimination récursive dans chaque orbite non triviale O de G des transitions dans $\mathrm{Out}(O) \times \mathrm{In}(O)$.

D'après le Lemme 2.16, le graphe sans orbite d'un graphe des positions est le graphe des positions de l'expression de départ, dont les opérateurs de clôture ont été supprimés. Si ce graphe sans orbite peut être réduit à un unique état, alors il est possible de remonter à une expression rationnelle dont l'automate des positions est isomorphe à l'automate de départ. Mais pour cela, il faut d'abord retrouver les sous-expressions de clôture en commençant par réduire les anciennes orbites :

Lemme 2.24. *Soient $G = (V, U)$ un graphe dont toute orbite non triviale est fortement stable et fortement transverse, O une orbite non triviale de G et G' le graphe sans orbite de G. Si G' est réductible, alors O peut être réduite à un unique état par itération des règles R_1, R_2 et R_3 sur G', à la condition que les règles R_1 et R_2 soient uniquement appliquées à des couples d'états dans O^2 ou dans $(V \setminus O)^2$.*

Nous pouvons alors formuler la caractérisation des graphes des positions suivante :

Théorème 2.25. *Un graphe G est un graphe des positions si et seulement si les trois propriétés suivantes sont vérifiées :*

1. *G est un hamac,*
2. *toute orbite non triviale de G est fortement stable et fortement transverse,*
3. *le graphe sans orbite de G est réductible.*

Cette caractérisation peut être reformulée de manière à décrire l'algorithme calculant l'expression rationnelle finale. D'après le Lemme 2.15, toute orbite non triviale d'un graphe des positions est liée à une sous-expression de clôture maximale E. Et d'après le Lemme 2.16, il est possible d'en déduire le graphe des positions de l'expression E privée de sa clôture :

Définition 2.26. Soient G un graphe, O une orbite stable de G, et U_O l'ensemble des transitions entre états de O dans G moins celles dans $\mathrm{Out}(O) \times \mathrm{In}(O)$. Le *graphe déstabilisé de O* est le graphe (V, U) tel que :

— $V = O \cup \{i, f\}$,

— $U = U_O \cup \{i\} \times \mathrm{In}(O) \cup \mathrm{Out} \times \{f\}$.

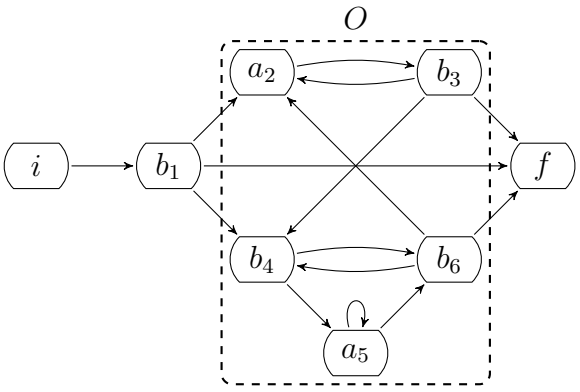


FIGURE 2.11 – Le graphe des positions de $b(ab + ba^*b)^*$

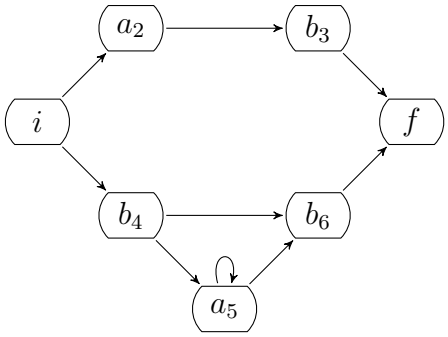


FIGURE 2.12 – Le graphe déstabilisé de l'orbite O

Enfin, d'après le Lemme 2.24, tous les ensembles d'états formant les orbites non triviales doivent pouvoir être réduits chacun à un unique état dans le graphe sans orbite. Le graphe restant doit à son tour être réductible pour que le graphe de départ soit caractérisé comme un graphe des positions. Cela revient à calculer, à partir du graphe des positions de départ, un graphe dans lequel les orbites, triviales ou non, sont représentées chacune par un unique état. Ce graphe se définit de la manière suivante :

Définition 2.27. Soient $G = (V, U)$ un graphe et $\mathrm{O}(G)$ l'ensemble des orbites de G. Le *graphe des orbites de* G est le graphe (V_O, U_O) tel que :

— $V_O = \mathrm{O}(G)$,

— $U_O = \{(O_1, O_2) \mid U^{\mathrm{out}}(O_1) \cap U^{\mathrm{in}}(O_2) \neq \emptyset\}$.

Le Théorème 2.25 peut alors être reformulé de la façon suivante :

Théorème 2.28. *Un graphe G est un graphe des positions si et seulement si les trois propriétés suivantes sont vérifiées :*

1. *G est un hamac,*
2. *toute orbite non triviale de G est stable et transverse, et son graphe déstabilisé est un graphe des positions,*
3. *le graphe des orbites de G est réductible.*

Ce Théorème décrit un algorithme récursif calculant une expression rationnelle correspondant à un graphe des positions G. Si G est acyclique, il est réduit et l'expression rationnelle calculée est retournée. Sinon, une orbite non triviale O est sélectionnée, et sa sous-expression de clôture E^+ est calculée en utilisant récursivement l'algorithme sur le graphe déstabilisé de O. La clôture positive est utilisée systématiquement car les expressions $H^+ + \varepsilon$ (créée par la règle R_3) et H^* produisent le même automate des positions, tout comme les expressions $H^+ + I$ (créée par la règle R_2) et $H^* + I$, tel que ε est dans $\mathrm{L}(I)$. Le graphe G' est ensuite crée en remplaçant tous les états de O dans G par un unique état E^+ ayant pour prédécesseurs (respectivement successeurs) directs les prédécesseurs (respectivement successeurs) directs de O. L'algorithme est alors utilisé récursivement sur G'.

Cet algorithme se termine toujours car le cas d'un graphe acyclique réductible se présente nécessairement : soit lorsque l'on traite le graphe déstabilisé d'une des sous-orbites non triviales les plus profondes, soit lorsque toutes les orbites non-triviales ont été testées et remplacées chacune par un unique état pour obtenir un graphe isomorphe au graphe des orbites. L'expression rationnelle obtenue a pour alphabet l'ensemble des états du graphe des positions. L'automate de départ étant homogène, il suffit alors de remplacer les états dans l'expression par la lettre des transitions arrivant dans celui-ci, à l'exception des états i et f qui sont remplacés par ε. Remarquons que l'expression obtenue est en forme normale de clôture.

Pour terminer, cette caractérisation permet de déduire celle des automates des positions construits à partir d'expressions rationnelles sans clôture positive. Toute orbite non triviale est alors liée à une sous-expression étoilée, dont le langage dénoté contient le mot vide. Soient E l'expression rationnelle retournée par l'algorithme précédemment décrit et F une sous-expression de clôture positive de E. Il faut alors vérifier que F est une sous-expression d'une sous-expression de somme $H = F + H_1 + H_2 + \cdots$ tel que ε est dans $\mathrm{L}(H)$. Si toutes les sous-expressions de clôture positive vérifient cette condition, alors il existe une expression équivalente sans clôture positive, obtenue en remplaçant les clôtures positives par des étoiles de Kleene, et ayant le même automate des positions.

Chapitre 3

Rappels des travaux précédents

Dans ce Chapitre, nous présentons deux sous-familles des langages rationnels : celle des langages non-ambigus et celle des langages déterministes. Ces langages peuvent être dénotés par des expressions rationnelles dont les automates des positions présentent une propriété structurelle, en l'occurrence la non-ambiguïté ou le déterminisme.

3.1 Langages non-ambigus

Nous commençons par définir la notion d'ambiguïté sur un automate, et en étudier la décidabilité. Puis nous l'étendons aux expressions rationnelles, et montrons que tout langage rationnel peut être dénoté par une telle expression.

3.1.1 Automates non-ambigus

Le *degré d'ambiguïté d'un mot* w dans un automate est égal au nombre de chemins réussis distincts ayant w pour étiquette. Le *degré d'ambiguïté d'un automate* [56] se définit alors comme le degré maximal d'un mot parmi ceux du langage reconnu. Ce maximum n'existe pas nécessairement, auquel cas l'automate est dit *infiniment ambigu*, avec la distinction suivante :

— il est *polynomialement ambigu* s'il existe un polynôme h tel que pour tout mot w, le degré d'ambiguïté de w y est inférieur ou égal à $h(|w|)$.

— sinon, il est *exponentiellement ambigu*.

Exemple 3.1. Étudions le degré d'ambiguïté des mots a^n pour tout entier n dans les automates A_1 et A_2 représentés en Figure 3.1. Il est de $n+1$ dans A_1 qui est donc polynomialement ambigu, et de 2^n dans A_2 qui est donc exponentiellement ambigu.

FIGURE 3.1 – Exemples d'automates infiniment ambigus

Weber et Seidl [56] montrent la décidabilité de ces propriétés :

Théorème 3.2 ([56])**.** *Soit A un automate émondé. Alors :*

1. *A est infiniment ambigu si et seulement s'il existe deux états distincts p et q de A et un mot non vide u tel que (p, u, p), (p, u, q) et (q, u, q) sont dans δ_A^t ;*
2. *A est exponentiellement ambigu si et seulement s'il existe un état p et deux chemins distincts bouclant sur p par le même mot.*

FIGURE 3.2 – Illustrations du Théorème 3.2

Nous nous intéressons à présent à un cas particulier d'automate finiment ambigu reconnaissant chaque mot de manière unique. Remarquons que selon notre définition d'un chemin dans un automate, le degré d'ambiguïté du mot vide est toujours inférieur ou égal à 1, même s'il existe plusieurs possibilités de reconnaître le mot vide. Nous posons donc une condition supplémentaire sur l'intersection de l'ensemble des états initiaux avec celui des finaux :

Définition 3.3. Un automate A est *non-ambigu* s'il a au plus un état à la fois initial et final et si son degré d'ambiguïté est inférieur ou égal à 1.

L'exemple le plus simple est celui des AFD : ils ont un unique état initial et aucun embranchement pour une même lettre.

Lemme 3.4. *Un automate déterministe est non-ambigu.*

Cette unicité dans les étiquettes des chemins de l'automate permet d'énoncer une condition nécessaire et suffisante à l'existence de cette propriété :

Proposition 3.5. *Soit A un automate émondé. Alors A est non-ambigu si et seulement si les deux conditions suivantes sont vérifiées :*

1. *pour tout deux états initiaux distincts i et j de A, l'intersection de $\mathrm{L}(i)$ avec $\mathrm{L}(j)$ est vide ;*
2. *pour toutes deux transitions distinctes (p, a, q) et (p, a, r) de A, l'intersection de $\mathrm{L}(q)$ avec $\mathrm{L}(r)$ est vide.*

Démonstration. Si la condition (1) n'est pas vérifiée, alors A n'est pas non-ambigu. Supposons maintenant que la condition (2) ne soit pas vérifiée. Il existerait alors un mot v dans l'intersection de $\mathrm{L}(q)$ avec $\mathrm{L}(r)$. Comme A est émondé, alors il existerait également un chemin étiqueté par un mot u allant d'un état initial vers p. Le mot uav serait donc l'étiquette de deux chemins réussis de A, qui ne serait pas non-ambigu.

Supposons maintenant que les deux conditions sont vérifiées. D'après la condition (1), il existe au plus un état à la fois initial et final. Soit w un mot non vide étiquetant deux chemins réussis c_1 et c_2 de A. D'après la condition (1), ces deux chemins partent du même état initial. Et d'après la condition (2), à chaque choix de transition pour les deux chemins, il n'y a qu'un seul successeur direct permettant de lire le restant de w. Donc c_1 et c_2 sont égaux et A est non-ambigu. ■

Ce résultat permet d'envisager un test de la non-ambiguïté d'un automate à partir du calcul des intersections des langages droits. Pour cela, nous avons recours à une construction utilisée notamment pour calculer un automate reconnaissant l'intersection des langages reconnus par deux automates.

Définition 3.6. *L'intersection de deux automates A* et *B* est l'automate $A \cap B = (\Sigma_A \cap \Sigma_B,\, Q_A \times Q_B,\, I_A \times I_B,\, F_A \times F_B,\, \delta)$ avec $\delta = \{((p_A, p_B),\, a,\, (q_A, q_B)) \mid (p_A, a, q_A) \in \delta_A \wedge (p_B, a, q_B) \in \delta_B\}$.

Proposition 3.7. *Soient A et B deux automates. Alors :*

— $\mathrm{L}(A \cap B) = \mathrm{L}(A) \cap \mathrm{L}(B)$,

— *pour tout état (p, q) de $A \cap B$,* $\mathrm{L}_{A\cap B}((p, q)) = \mathrm{L}_A(p) \cap \mathrm{L}_B(q)$.

Démonstration. La définition de l'ensemble des transitions de $A \cap B$ implique que $((p_A, p_B), w, (q_A, q_B))$ est dans $\delta^t_{A\cap B}$ si et seulement si (p_A, w, q_A) est dans δ^t_A et (p_B, w, q_B) est dans δ^t_B. Puisque $F_{A\cap B} = F_A \times F_B$, alors w est dans $\mathrm{L}_{A\cap B}((p, q))$ si et seulement si w est à la fois dans $\mathrm{L}_A(p)$ et dans $\mathrm{L}_B(q)$. Et puisque $I_{A\cap B} = I_A \times I_B$, alors w est dans $\mathrm{L}(A \cap B)$ si et seulement si w est à la fois dans $\mathrm{L}(A)$ et dans $\mathrm{L}(B)$. ■

Le test de la non-ambiguïté d'un automate peut alors se résumer à un test sur la structure de son intersection avec lui-même :

Proposition 3.8. *Soient A un automate et A^2 la partie émondée de $A \cap A$. Alors A est non-ambigu si et seulement si tous les états de A^2 sont de la forme (p, p).*

Démonstration. Soit (p, q) un état de $A \cap A$ tel que p et q soient distincts. D'après la Proposition 3.7, (p, q) est co-accessible si et seulement s'il existe un mot commun au langage droit de p et de q, et accessible si et seulement s'il existe un mot u tel que p et q sont dans $\delta^t(I_A, u)$. Donc, d'après la Proposition 3.5, (p, q) est présent dans A^2 si et seulement si A est ambigu. ■

Ce qui permet de conclure que :

Théorème 3.9 ([1])**.** *La non-ambiguïté d'un automate A est décidable en temps $O(|\delta_A|^2)$.*

3.1.2 Expressions rationnelles non-ambiguës

Les résultats de cette Section proviennent de Book et al.[5].

Le *degré d'ambiguïté* d'un mot w peut également se définir dans une expression rationnelle, comme le nombre de mots u distincts appartenant au langage dénoté par sa linéarisée tel que $u^\natural = w$. Remarquons que le degré d'ambiguïté de ε dans une expression rationnelle est nécessairement inférieur ou égal à 1. Nous transposons alors la notion de non-ambiguïté aux expressions rationnelles :

Définition 3.10. Une expression rationnelle E est *non-ambiguë* si le degré d'ambiguïté de tout mot y est inférieur ou égale à 1. Un langage est *non-ambigu* s'il peut être dénoté par une expression rationnelle non-ambiguë.

Exemple 3.11. Soient les expressions $E = (a+b)^*aa^*$ et $F = (a+b)^*a$. Ces deux expressions sont équivalentes mais la seconde est non-ambiguë, tandis que la première ne l'est pas : le langage dénoté par l'expression $(a_1+b_2)^*a_3a_4^*$ contient les mots a_1a_3 et a_3a_4 dont les délinéarisés sont égaux.

Remarquons que la définition du degré d'ambiguïté d'un mot dans une expression et la construction de l'automate des positions impliquent la propriété suivante :

Lemme 3.12. *Un mot w a le même degré d'ambiguïté dans une expression rationnelle E et dans l'automate des positions de E.*

Démonstration. Soit E une expression rationnelle. D'après la Définition 1.9 et la Proposition 1.10, le langage dénoté par $E^\sharp$ est reconnu par l'automate $P_E^\sharp$. De plus, $P_E^\sharp$ est déterministe et donc non-ambigu. Comme l'automate P_E est obtenu en supprimant les indices des étiquettes des transitions de $P_E^\sharp$, l'ambiguïté de chaque mot dans P_E est la même que celle dans E. ■

La propriété de non-ambiguïté d'une expression rationnelle peut alors être liée à celle de son automate des positions :

Corollaire 3.13. *Une expression rationnelle est non-ambiguë si et seulement si son automate des positions est non-ambigu.*

Cette définition alternative permet donc de décider de la non-ambiguïté d'une expression rationnelle :

Théorème 3.14. *La non-ambiguïté d'une expression rationnelle E est décidable en temps $O(||E||^4 + |E|)$.*

Démonstration. Soient E une expression rationnelle et P son automate des positions. D'après le Théorème 2.9, P peut être construit en temps $O(||E||^2 + |E|)$. Comme $|\delta_P|$ est au plus égal à $|Q_P|^2$ et que $|Q_P| = ||E||$, le Théorème 3.9 permet de conclure. ■

Pour terminer, nous cherchons quels langages peuvent être dénotés par une telle expression. Nous étudions alors l'expression rationnelle obtenue par l'algorithme de Brzozowski et Mc-Cluskey sur un automate intermédiaire dit *primitif*, dont toutes les transitions ont une étiquette qui leur est propre.

Proposition 3.15 ([5])**.** *L'expression rationnelle obtenue par l'algorithme de Brzozowski et Mc-Cluskey sur un automate primitif est non-ambiguë.*

Démonstration. Soit A un automate primitif et $A' = (\Sigma_A, Q_A \cup \{i, f\}, \{i\}, \{f\}, \delta_A \cup \delta_i \cup \delta_f)$ l'automate standard et co-standard obtenu au départ de l'algorithme de Brzozowski et Mc-Cluskey.

Pour tout deux états p et q de A', l'expression $E_{p,r}^0$ est composée d'un seul symbole (une lettre ou ε) et est donc non-ambiguë. Supposons maintenant que toutes les expressions obtenues après élimination des $k-1$ premiers états sont non-ambiguës.
Le langage dénoté par $E_{p,r}^{k-1}$ correspond à l'ensemble des étiquettes des chemins entre p et r dans A' ne traversant que des états strictement inférieurs à q_k, tandis que celui dénoté par $E_{p,q_k}^{k-1}(E_{q_k,q_k}^{k-1})^* E_{q_k,r}^{k-1}$ correspond à l'ensemble des étiquettes des chemins entre p et r dans A' traversant obligatoirement q_k mais sans traverser les états qui lui sont strictement supérieurs. L'automate A étant primitif, tous les chemins y ont une étiquette qui leur est

propre. Donc l'intersection de $\mathrm{L}(E_{p,r}^{k-1})$ et de $\mathrm{L}(E_{p,q_k}^{k-1}(E_{q_k,q_k}^{k-1})^* E_{q_k,r}^{k-1})$, privée de ε, est vide. Comme $E_{p,r}^{k-1}$ est non-ambiguë (par hypothèse d'induction), il ne reste plus qu'à démontrer que $E_{p,q_k}^{k-1}(E_{q_k,q_k}^{k-1})^* E_{q_k,r}^{k-1}$ l'est aussi.
Les étiquettes non vides des chemins dénotées par E_{p,q_k}^{k-1} se terminent toutes par l'étiquette d'une transition arrivant en q_k et ne contiennent aucune autre étiquette de transition impliquant q_k. Les étiquettes des chemins dénotées par $(E_{q_k,q_k}^{k-1})^* E_{q_k,r}^{k-1}$ commencent elles toutes par une transition sortant de q_k. L'automate A étant primitif, le seul suffixe de $\mathrm{L}(E_{p,q_k}^{k-1})$ qui soit commun avec un préfixe de $\mathrm{L}((E_{q_k,q_k}^{k-1})^* E_{q_k,r}^{k-1})$ est le mot vide. De même, les étiquettes non vides des chemins dénotées par $E_{q_k,r}^{k-1}$ commencent toutes par une transition sortant de q_k et ne contiennent aucune autre transition impliquant q_k, tandis que celles dénotées par $E_{p,q_k}^{k-1}(E_{q_k,q_k}^{k-1})^*$ se terminent toutes par une transition arrivant en q_k. Là aussi, le seul préfixe de $\mathrm{L}(E_{q_k,r}^{k-1})$ qui soit commun avec un suffixe de $\mathrm{L}(E_{p,q_k}^{k-1}(E_{q_k,q_k}^{k-1})^*)$ est le mot vide. Donc pour tout mot w dénoté par $E_{p,q_k}^{k-1}(E_{q_k,q_k}^{k-1})^* E_{q_k,r}^{k-1}$, il existe une unique décomposition $w_p w_f w_s$ tel que w_p est dénoté par E_{p,q_k}^{k-1}, w_f par $(E_{q_k,q_k}^{k-1})^*$ et w_s par $E_{q_k,r}^{k-1}$. Par hypothèse d'induction, ces trois sous-expressions sont non-ambiguës et donc $E_{p,q_k}^{k-1}(E_{q_k,q_k}^{k-1})^* E_{q_k,r}^{k-1}$ l'est aussi. ■

Or, pour tout automate A, il est possible de créer un automate primitif reconnaissant l'ensemble des chemins réussis de A, et de se servir de l'expression rationnelle dénotant le langage reconnu pour construire une expression dénotant le langage reconnu par A.

Théorème 3.16. *Un mot w a le même degré d'ambiguïté dans un automate et dans l'expression rationnelle obtenue à partir de celui-ci par l'algorithme de Brzozowski et Mc-Cluskey.*

Démonstration. Soit A un automate. L'automate $A_t = (\Sigma_A, Q_A, I_A, F_A, \delta)$ tel que $\delta = \{(p,(p,a,q),q) \mid (p,a,q) \in \delta_A\}$ est primitif et reconnaît l'ensemble des chemins réussis de A.

D'après la Proposition 3.15, l'algorithme de Brzozowski et Mc-Cluskey appliqué à A_t produit une expression rationnelle E_t non-ambiguë dénotant tous les chemins de A.

Si tous les symboles alphabétiques (p,a,r) de E_t sont remplacés par a, alors l'expression rationnelle E obtenue dénote le langage reconnu par A tel que tous les mots ont le même degré d'ambiguïté dans E et dans A. ■

Enfin, d'après le Théorème 1.30 et le Lemme 3.4, tout langage rationnel est reconnaissable par un automate non-ambigu. Autrement dit :

Théorème 3.17 ([5])**.** *Tout langage rationnel est non-ambigu.*

3.2 Langages déterministes

3.2.1 Expressions rationnelles déterministes

Le standard ISO du SGML [40] utilise des expressions rationnelles ayant pour propriété que tout mot w du langage dénoté doit avoir une unique séquence de positions dans l'expression, tel que chaque position se déduit de la précédente et du prochain symbole à lire dans w. Ce qui se traduit par la Définition suivante :

Définition 3.18 ([14])**.** Une expression rationnelle E est *déterministe* si pour tout mot u, v et w sur Π_E et toutes deux positions distinctes x et y tel que uxv et uyw sont des mots de $\mathrm{L}(E^\sharp)$, alors $x^\natural$ est différent de $y^\natural$. Un langage est *déterministe* s'il peut être dénoté par une expression rationnelle déterministe.

D'après les définitions des fonctions First et Follow, nous obtenons le résultat suivant :

Lemme 3.19 ([14])**.** *Une expression rationnelle E est déterministe si et seulement si les deux conditions suivantes sont vérifiées :*

1. *pour toutes deux positions distinctes x et y dans* $\mathrm{First}(E^\sharp)$*, $x^\natural$ est différent de $y^\natural$,*
2. *pour tout deux couples distincts (x, y) et (x, z) dans* $\mathrm{Follow}(E^\sharp)$*, $y^\natural$ est différent de $z^\natural$.*

Corollaire 3.20 ([14])**.** *Une expression rationnelle E est* déterministe *si et seulement si son automate des positions est déterministe.*

Exemple 3.21. Soient les expressions $E = (a + b)^*a$ et $F = b^*a(b^*a)^*$. Les automates des positions de E et F sont représentés respectivement en Figure 3.3 et en Figure 3.4. Ces deux expressions sont équivalentes et non-ambiguës. Mais d'après le Corollaire 3.20, seule la seconde est déterministe.

Cette équivalence avec une propriété structurelle de l'automate des positions implique que tout langage déterministe peut être dénoté par une expression rationnelle déterministe présentant quelques unes des propriétés étudiées précédemment. Tout d'abord, d'après le Théorème 2.7, la mise en forme normale étoilée préserve l'automate des positions. Ensuite, d'après le Lemme 2.3, l'émondage d'une expression rationnelle produit un sous-automate de l'automate des positions de départ. Si ce dernier est déterministe, sa partie émondée l'est donc également. Nous en déduisons que :

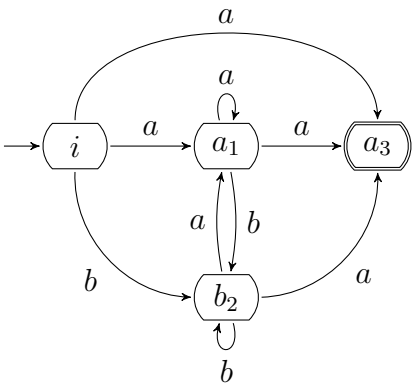


FIGURE 3.3 – $(a+b)^*a$

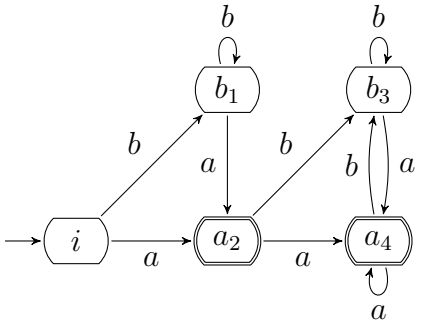


FIGURE 3.4 – $b^*a(b^*a)^*$

Lemme 3.22 ([14])**.** *Tout langage déterministe peut être dénoté par une expression rationnelle déterministe, émondée et en forme normale étoilée.*

Nous utilisons alors les propriétés de la forme normale étoilée afin de caractériser le déterminisme d'une expression rationnelle de manière inductive :

Lemme 3.23 ([14])**.** *Soit E une expression rationnelle en forme normale étoilée.*

- *Si E est égale à $\emptyset$, ε ou une lettre a, alors E est déterministe ;*
- *Si $E = F + G$, alors E est déterministe si et seulement si F et G sont déterministes et* $\mathrm{First}(F) \cap \mathrm{First}(G) = \emptyset$ *;*
- *Si $E = FG$, alors E est déterministe si et seulement si F et G sont déterministes,* $\mathrm{Followlast}(F) \cap \mathrm{First}(G) = \emptyset$*, et si* $\mathrm{Null}(F)$ *est vérifiée alors* $\mathrm{First}(F) \cap \mathrm{First}(G) = \emptyset$ *;*
- *Si $E = F^*$, alors E est déterministe si et seulement si F est déterministe et* $\mathrm{Followlast}(F) \cap \mathrm{First}(F) = \emptyset$.

Démonstration. Tout d'abord, remarquons que pour qu'une expression soit déterministe, il est nécessaire que toutes ses sous-expressions le soient aussi. Nous supposons donc dans le reste de cette preuve que toutes les sous-expressions distinctes de E sont déterministes. Nous utiliserons également le Lemme 3.19 dont nous mentionnons les deux conditions par (1) et (2).

Si $E = F + G$ alors $\mathrm{Follow}(E^\sharp) = \mathrm{Follow}(F^\sharp) \cup \mathrm{Follow}(G^\sharp)$. Étant donné que F et G ne partagent aucune position, (2) est vérifiée. De plus, l'ensemble $\mathrm{First}(E^\sharp) = \mathrm{First}(F^\sharp) \cup \mathrm{First}(G^\sharp)$ étant une union disjointe, (1) est vérifiée si et seulement si l'intersection de $\mathrm{First}(F)$ avec $\mathrm{First}(G)$ est vide.

Si $E = FG$, alors $\mathrm{Follow}(E^\sharp) = \mathrm{Follow}(F^\sharp) \cup \mathrm{Follow}(G^\sharp) \cup \mathrm{Last}(F^\sharp) \times \mathrm{First}(G^\sharp)$. Les seules positions nous intéressant sont celles dans $\mathrm{Last}(F^\sharp)$, qui peuvent être suivies à la fois par des positions de F et de G. Par définition de la fonction Followlast, (2) est alors vérifiée si et seulement si l'intersection de $\mathrm{Followlast}(F)$ avec $\mathrm{First}(G)$ est vide.
Si $\mathrm{Null}(F)$ n'est pas vérifiée, alors $\mathrm{First}(E^\sharp) = \mathrm{First}(F^\sharp)$ et (1) est vérifiée. Si $\mathrm{Null}(F)$ est vérifiée, alors $\mathrm{First}(E^\sharp) = \mathrm{First}(F^\sharp) \cup \mathrm{First}(G^\sharp)$. Cette union étant disjointe, (1) est donc vérifiée si et seulement si l'intersection de $\mathrm{First}(F)$ avec $\mathrm{First}(G)$ est vide.

Si $E = F^*$, alors $\mathrm{First}(F^*) = \mathrm{First}(F)$ et (1) est vérifiée. Comme E est en forme normale étoilée, alors l'ensemble $\mathrm{Follow}(E^\sharp) = \mathrm{Follow}(F^\sharp) \cup \mathrm{Last}(F^\sharp) \times \mathrm{First}(F^\sharp)$ est une union disjointe. Les seules positions nous intéressant sont celles dans $\mathrm{Last}(F^\sharp)$, qui peuvent être suivies à la fois par des positions dans $\mathrm{Followlast}(F^\sharp)$ et dans $\mathrm{First}(F^\sharp)$. Donc (2) est vérifiée si et seulement si l'intersection de $\mathrm{Followlast}(F)$ avec $\mathrm{First}(F)$ est vide. ■

En utilisant les propriétés de la forme normale étoilée énoncées au Théorème 2.9, et la caractérisation que nous venons de décrire, il est possible de tester le déterminisme d'une expression et de construire son automate des positions en temps linéaire. En effet, chaque position d'une expression déterministe est suivie par un nombre de positions au plus égal au cardinal de l'alphabet de l'expression.

Théorème 3.24 ([13, 33])**.** *Le déterminisme d'une expression rationnelle E est décidable en temps $O(|E|)$. Si E est déterministe, alors son automate des positions est calculable en temps $O(|E|)$.*

Une autre propriété découlant du déterminisme et de la forme normale étoilée concerne son ensemble des dérivées. D'après la caractérisation ci-dessus, les formules de dérivation se simplifient de la manière suivante :

Lemme 3.25. *Soient E une expression rationnelle déterministe et a une lettre de son alphabet.*

— *Si $E = F + G$, alors*

$$\mathrm{d}_a(E) = \begin{cases} \mathrm{d}_a(F) & \textit{si } a \in \mathrm{First}(F) \\ \mathrm{d}_a(G) & \textit{sinon} \end{cases}$$

— *Si $E = F \cdot G$, alors*

$$\mathrm{d}_a(E) = \begin{cases} \mathrm{d}_a(F) \cdot G & \textit{si } a \in \mathrm{First}(F) \\ \mathrm{d}_a(G) & \textit{si } a \in \mathrm{First}(G) \wedge \mathrm{Null}(F) \\ \emptyset & \textit{sinon} \end{cases}$$

Ces formules permettent de montrer que les dérivées d'expressions rationnelles déterministes en forme normale étoilée sont elles-mêmes déterministes :

Théorème 3.26 ([14]). *Soit E une expression rationnelle sur un alphabet Σ. Si E est déterministe et en forme normale étoilée, alors pour toute lettre a de Σ, $\mathrm{d}_a(E)$ est déterministe.*

Démonstration. Remarquons d'abord que pour toute expression H, l'ensemble Followlast($\mathrm{d}_a(H)$) est un sous-ensemble de Followlast(H), et que si ε est dans le langage dénoté par $\mathrm{d}_a(H)$, alors l'ensemble First($\mathrm{d}_a(H)$) est un sous-ensemble de Followlast(H).

Si E vaut $\emptyset$, ε ou une lettre b, alors $\mathrm{d}_a(E)$ est déterministe.

Si $E = F + G$, ou si $E = F \cdot G$ et a n'est pas dans First(F), alors d'après le Lemme 3.25 et par hypothèse d'induction, $\mathrm{d}_a(E)$ est déterministe.

Si maintenant $E = F \cdot G$ et a est dans First(F), alors d'après le Lemme 3.25, $\mathrm{d}_a(E) = \mathrm{d}_a(F) \cdot G$. Par hypothèse d'induction, $\mathrm{d}_a(F)$ est déterministe. Comme E est déterministe, G l'est aussi et Followlast(F) $\cap$ First(G) est vide. D'après la première remarque, nous obtenons que Followlast($\mathrm{d}_a(F)$) $\cap$ First(G) est vide. Et si Null($\mathrm{d}_a(F)$) est vérifiée, alors d'après la seconde remarque, First($\mathrm{d}_a(F)$) $\cap$ First(G) est vide. Donc $\mathrm{d}_a(E)$ est déterministe.

Enfin, si $E = F^*$, alors $\mathrm{d}_a(E) = \mathrm{d}_a(F) \cdot F^*$. Par hypothèse d'induction, $\mathrm{d}_a(F)$ est déterministe. De plus, E étant en forme normale étoilée, alors Followlast(F) $\cap$ First(F^*) est vide. D'après la première remarque, Followlast($\mathrm{d}_a(F)$) $\cap$ First(F^*) l'est aussi. Et si ε est dans le langage dénoté par $\mathrm{d}_a(F)$, alors First($\mathrm{d}_a(F)$) $\cap$ First(F^*) est également vide. Donc $\mathrm{d}_a(E)$ est déterministe. ■

Remarquons alors que d'après la Table 1.3 de calcul des dérivées, toutes les sous-expressions étoilées de la dérivée d'une expression rationnelle E sont des sous-expressions étoilées de E. Autrement dit :

Lemme 3.27. *Soit E une expression rationnelle sur un alphabet Σ. Si E est en forme normale étoilée, alors pour toute lettre a de Σ, $\mathrm{d}_a(E)$ est en forme normale étoilée.*

Chen *et al.* utilisent ces résultats pour montrer la finitude de l'ensemble des dérivées de ces expressions du Théorème 1.22, mais sans avoir recours au quotientage par les relations d'équivalence sur l'opérateur + :

Théorème 3.28 ([24]). *Soit E une expression rationnelle. Si E est déterministe et en forme normale étoilée, alors le cardinal de $\mathcal{D}(E)$ est inférieur ou égal à $||E|| + 1$.*

3.2.2 La famille des langages déterministes

D'après le Théorème 1.30, tout langage rationnel est reconnaissable par un automate déterministe. Mais la question est ici de savoir si tout langage rationnel est reconnaissable par un automate des positions déterministe. Nous commençons par présenter deux propriétés de cette famille.

La première concerne les résiduels de ces langages. En effet, la dérivée d'une expression rationnelle dénote un résiduel du langage dénoté par cette dernière. Alors, en prenant pour point de départ le Lemme 3.22, et en appliquant le Théorème 3.26 et le Lemme 3.27, nous obtenons que :

Théorème 3.29 ([14])**.** *Si un langage est déterministe, alors tous ses résiduels le sont.*

La seconde concerne l'inclusion des langages finis :

Théorème 3.30. *Tout langage rationnel fini est déterministe.*

Démonstration. Soit L un langage rationnel fini. Nous montrons que L est déterministe par récurrence sur la longueur k de son plus long mot. Si $k = 0$, alors L vaut $\emptyset$ ou $\{\varepsilon\}$ qui sont tous les deux déterministes. Sinon, d'après le Lemme 1.14, L vaut soit $\{\varepsilon\} \cup \bigcup_{a\in\Sigma}(\{a\} \cdot a^{-1}L)$ soit $\bigcup_{a\in\Sigma}(\{a\} \cdot a^{-1}L)$. Pour toute lettre a, la longueur du plus long mot de $a^{-1}L$ est au plus de $k - 1$ et, par hypothèse de récurrence, $a^{-1}L$ est déterministe. Donc, d'après les Lemmes 1.20 et 3.23, il est possible de construire une expression rationnelle déterministe reconnaissant L dans les deux cas. ■

Nous nous intéressons alors au cas général des langages rationnels infinis. Par définition, si un tel langage est déterministe, alors son AFD minimal peut être calculé en minimisant un automate des positions déterministe, contenant donc des orbites non triviales. Orbites qui, d'après le Lemme 2.15, sont fortement stables et fortement transverses. Nous étudions donc si ces propriétés sont préservées par minimisation, afin de pouvoir tester l'appartenance à la famille sur un représentant canonique du langage.

Exemple 3.31. L'automate des positions P de l'expression rationnelle $(aa+bb)(b(\varepsilon+(\varepsilon+c)a))^*d$ est représenté en Figure 3.5 et son AFD minimal M en Figure 3.6. L'orbite $O = \{b_5, c_6, a_7\}$ dans P est telle que $\Omega(O) = K$ avec $K = \{\{a_1, c_6\}, \{a_2, b_4, a_7\}, \{b_5\}\}$. Nous remarquons que l'orbite K de M ne présente pas les propriétés énoncées en Section 2.3 : ses portes d'entrée n'ont pas les mêmes prédécesseurs en dehors de l'orbite, et ses portes de sortie n'ont pas de transition vers une même porte d'entrée. Cependant, ses portes de sortie ont bien le même successeur par d en dehors de l'orbite, et ont toutes une transition étiquetée par b vers le même état dans l'orbite.

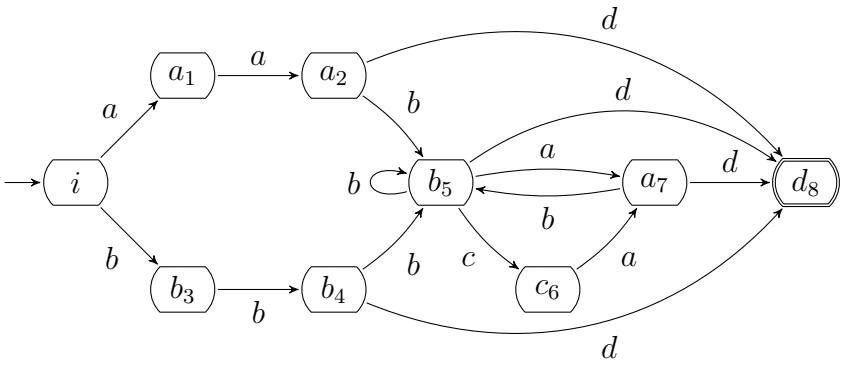


FIGURE 3.5 – $E = (aa + bb)(b(\varepsilon + (\varepsilon + c)a))^*d$

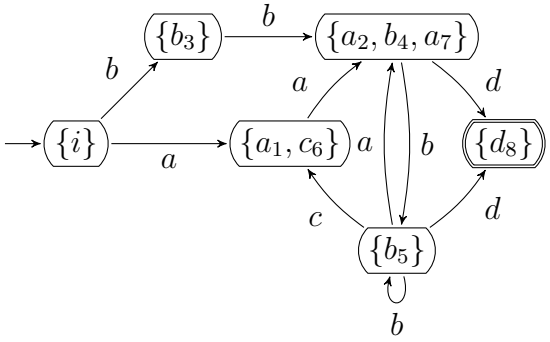


FIGURE 3.6 – L'AFD minimal de L(E)

Dans l'exemple précédent, nous constatons que les propriétés définies sur les portes d'entrée ne sont pas préservées par minimisation. Ceci est dû au fait que si un état p est une porte d'entrée d'une orbite O, l'état $\Phi(p)$ n'est pas nécessairement une porte d'entrée de $\Omega(O)$. Nous étudions alors le cas des portes de sortie. D'après le Lemme 1.46, si O est maximale alors elle contient au moins un antécédent de chaque état de l'orbite $\Omega(O)$. Nous utilisons alors les relations structurelles entre un AFD et son minimal, étudiées au Lemme 1.44, pour montrer que les classes d'équivalence des portes de sortie d'une orbite maximale sont aussi des portes de sortie :

Lemme 3.32. *Soient A un AFD, M son AFD minimal et K une orbite de M. Alors pour toute orbite maximale O de A tel que $\Omega(O) = K$, nous avons :*

1. $\delta^{\text{out}}(K) = \{(\Phi(g), a, \Phi(q)) \mid (g, a, q) \in \delta^{\text{out}}(O)\}$,
2. $\Phi(\text{Out}(O)) = \text{Out}(K)$.

Démonstration. Soit O une orbite maximale de A tel que $\Omega(O) = K$.

Soit (g, a, q) une transition de sortie de O dans A. D'après le Lemme 1.44, $(\Phi(g), a, \Phi(q))$ est dans δ_M. De plus, $\Phi(q)$ n'est pas dans K car q est accessible depuis O sans lui appartenir alors que O est maximale. Comme $\Phi(g)$ est dans K, alors $(\Phi(g), a, \Phi(q))$ est une transition de sortie de K.
Soit (g', a, q') une transition de sortie de K dans M. D'après le Lemme 1.44, pour tout état g dans $\Phi^{-1}(g')$, il existe un état q dans $\Phi^{-1}(q')$ tel que (g, a, q) est dans δ_A. Puisque O est maximale, d'après le Lemme 1.46, l'état q n'est pas dans O car q' n'est pas dans K, et l'état g est dans O car g' est dans K. Donc (g, a, q) est une transition de sortie de O.

D'après le Lemme 1.46, nous avons $\Phi(O \cap F_A) = K \cap F_M$. Le premier point permet de conclure pour le cas des portes non finales. ■

Nous présentons donc quelques propriétés structurelles liées aux portes de sortie des orbites d'un AFD, et ce qu'elles impliquent dans le cas des automates des positions. Celles-ci permettront d'énoncer un test dont nous montrons que la validation par un AFD minimal est une condition nécessaire et suffisante pour décider si un langage est déterministe.

3.2.3 La décidabilité du déterminisme

3.2.3.1 Transversalité et langages orbitaux

La transversalité d'une orbite d'un automate s'inspire donc de la propriété de transversalité en sortie d'une orbite d'un graphe. Cette dernière implique que toutes les portes de sortie ont les mêmes connexions avec tout élément externe à l'orbite. Dans le cas d'un automate, cela implique de prendre en compte les étiquettes des transitions et la finalité des états :

Définition 3.33. Une orbite O d'un automate est *transverse* si pour toutes deux portes de sortie s_1 et s_2 de O, s_2 est finale si s_1 l'est et (s_2, a, r) est une transition de sortie de O si (s_1, a, r) l'est. Un automate satisfait la *propriété orbitale* si toutes ses orbites sont transverses.

Si une orbite est transverse, nous pouvons alors définir son *langage externe*, comme l'ensemble $\mathrm{L}^{\mathrm{ext}}(O)$ des mots étiquetant les chemins allant d'une porte de sortie de O vers un état final sans traverser d'état de O (cela inclus le chemin vide si les portes sont finales). À l'inverse, nous nous intéressons maintenant au langage constitué des mots étiquetant les chemins allant d'un état vers les portes de sortie de son orbite, et définissons un nouvel automate pour le reconnaître :

Définition 3.34. Soient A un automate et q un état de A. L'*automate orbital de q* est l'automate A_q, obtenu en conservant uniquement les états de $\mathrm{O}_A(q)$, et ayant pour état initial q et pour états finaux les portes de $\mathrm{O}_A(q)$. Le langage reconnu par A_q est appelé le *langage orbital de q*, noté $\mathrm{L}^{\mathrm{orb}}(q)$, et fait partie de la famille des *langages orbitaux de A*.

Dans le cas des automates des positions, nous avons vu en Section 2.3 que les orbites non triviales sont liées aux sous-expressions étoilées maximales de l'expression de départ. Si l'expression est en plus déterministe, les langages orbitaux présentent les propriétés suivantes :

Lemme 3.35. *Soient E une expression rationnelle déterministe émondée et en forme normale étoilée, et P son automate des positions. Alors :*

1. *pour toute orbite non triviale O de P, il existe une sous-expression étoilée maximale H de E tel que les langages orbitaux des états de O sont des résiduels du langage dénoté par H ;*
2. *les langages orbitaux de P sont déterministes.*

Démonstration. Soit O une orbite non triviale de P. D'après le Lemme 2.15, il existe donc une sous-expression étoilée maximale H de E tel que tous les états de O sont des positions de $H^\sharp$. Soit P' l'automate des positions de H, qui est donc composé de deux orbites : une triviale contenant son état initial, et O qui n'a ici aucune transition de sortie. En effet, d'après le Lemme 2.15, les états de O, ses transitions internes, et ses portes d'entrée et de sortie sont présents à l'identique dans P'. Les langages orbitaux des états de O dans P sont donc les mêmes que leurs langages droits dans P'. Or, E étant déterministe, H l'est aussi et les langages droits de tous les états de P' sont des résiduels de $\mathrm{L}(H)$.

En appliquant le Théorème 3.29, nous concluons que les langages orbitaux des orbites non triviales sont déterministes. Quant aux orbites triviales, elles partagent le langage orbital $\{\varepsilon\}$ qui est également déterministe. ■

Le langage externe de l'orbite représente, quant à lui, le langage dénoté par les sous-expressions suivant la sous-expression étoilée maximale dans un produit de concaténation. Cette distinction dans les chemins internes et externes à une orbite permet de décomposer les langages droits des états d'une orbite transverse comme suit :

Lemme 3.36. *Soit O une orbite transverse d'un automate A. Alors :*

1. *pour tout état q de O, nous avons* $\mathrm{L}(q) = \mathrm{L}^{\mathrm{orb}}(q) \cdot \mathrm{L}^{\mathrm{ext}}(O)$,
2. $\mathrm{L}^{\mathrm{ext}}(O) = \begin{cases} \{\varepsilon\} \cup \bigcup_{(g,a,p)\in\delta^{\mathrm{out}}(O)}(\{a\}\cdot \mathrm{L}(p)) & \textit{si } \mathrm{Out}(O)\cap F_A \neq \emptyset \\ \bigcup_{(g,a,p)\in\delta^{\mathrm{out}}(O)}(\{a\}\cdot \mathrm{L}(p)) & \textit{sinon} \end{cases}$.

Démonstration. Le second point est une conséquence directe de la définition du langage externe d'une orbite transverse.

Soit u un mot dans $\mathrm{L}^{\mathrm{orb}}(q)$ et v un mot dans $\mathrm{L}^{\mathrm{ext}}(O)$. Il existe donc une porte g de O et un chemin allant de q vers g étiqueté par u. Comme O est transverse, alors v appartient à $\mathrm{L}(g)$ et uv appartient à $\mathrm{L}(q)$.
Soit w un mot dans $\mathrm{L}(q)$. Il existe donc un état final f et un chemin acceptant c allant de p vers f étiqueté par w.
Si c ne contient que des transitions internes à O, alors f en est une porte de sortie car il est final. Donc w est dans $\mathrm{L}^{\mathrm{orb}}(q)$ et ε est dans $\mathrm{L}^{\mathrm{ext}}(O)$.
Sinon, $w = uv$ et c se partage en un chemin étiqueté par u et n'utilisant que des transitions internes à O jusqu'à l'une de ses portes de sortie, et un chemin étiqueté par v, commençant par une transition de sortie de O et allant vers un état final. Donc u est dans $\mathrm{L}^{\mathrm{orb}}(q)$ et v et dans $\mathrm{L}^{\mathrm{ext}}(O)$. ■

Ceci permet de donner une condition suffisante pour déterminer si le langage reconnu par un AFD est déterministe :

Proposition 3.37 ([14])**.** *Soit A un AFD émondé. Si A satisfait la propriété orbitale et que ses langages orbitaux sont déterministes, alors* $\mathrm{L}(A)$ *est déterministe et une expression rationnelle déterministe le dénotant peut être construite à partir de celles dénotant les langages orbitaux.*

Démonstration. Soit A un AFD satisfaisant la propriété orbitale tel que tous ses langages orbitaux soient déterministes.
Si A est composé d'une unique orbite, alors le langage reconnu par A est égal au langage orbital de son état initial i, qui est déterministe.
Supposons que A ait plusieurs orbites. Ces dernières étant transverses, d'après le Lemme 3.36, le langage reconnu par A est égal à $\mathrm{L}^{\mathrm{orb}}(i)\cdot\mathrm{L}^{\mathrm{ext}}(\mathrm{O}(i))$.
Soit $T = \{(a_1, q_1), (a_2, q_2), \cdots, (a_n, q_n)\}$ l'ensemble des couples étiquettes et états d'arrivées des transitions de sortie de $\mathrm{O}(i)$. Le langage externe de cette dernière est alors égal à $\{a_1\} \cdot \mathrm{L}(q_1) \cup \{a_2\} \cdot \mathrm{L}(q_2) \cup \cdots \cup \{a_n\} \cdot \mathrm{L}(q_n)$, en incluant ε si les portes sont finales. Ces langages $\mathrm{L}(q_x)$ sont donc reconnaissables par des AFD ayant moins d'orbites que A, satisfaisant la propriété orbitale et dont tous les langages orbitaux sont déterministes. Par hypothèse d'induction sur le nombre d'orbites, ces langages sont donc déterministes.
Comme A est déterministe et que $\mathrm{O}(i)$ est transverse, tous les symboles a_x sont distincts entre eux, et $\mathrm{L}^{\mathrm{ext}}(\mathrm{O}(i))$ peut être dénoté par une expression rationnelle déterministe E^{ext} tel que $\mathrm{First}(E^{\mathrm{ext}}) = \{a_1, a_2, \cdots, a_n\}$.
Ensuite, puisque les langages orbitaux sont déterministes, nous posons E^{orb} l'expression déterministe dénotant le langage orbital de i. Nous commençons par vérifier les propriétés de son ensemble Followlast. Comme A est

déterministe, aucune porte de sortie de O(i) n'a de transition interne étiquetée par a_1, a_2, $\cdots$, a_n. Donc aucun état final de l'automate orbital de i n'a de transition sortante étiquetée par a_1, a_2, $\cdots$, a_n, et l'intersection de Followlast(E^{orb}) avec First(E^{ext}) est vide. Si i est une porte, alors c'est un état final de son automate orbital et, par les mêmes arguments, l'intersection de First(E^{orb}) avec First(E^{ext}) est vide.
D'après le Lemme 3.23, l'expression $E^{\text{orb}} \cdot E^{\text{ext}}$ est donc déterministe. ■

3.2.3.2 Synchronisation et langages de coupure

Le résultat précédent ne nous permet pas de décider si les langages orbitaux des orbites non triviales sont déterministes. Nous nous inspirons alors de la propriété de stabilité pour définir une notion plus générale, basée sur l'existence de cibles communes à toutes les portes de sortie :

Définition 3.38. Soient A un automate, a une lettre et s un état de A. Le couple (a, s) est dit *A-synchronisant* si tous les états finaux de A ont une transition sortante étiquetée par a vers s. L'ensemble des couples synchronisants de A est noté $\mathcal{S}_A$.

Cette Définition par rapport aux états finaux se justifie par le fait que les portes de sortie d'une orbite sont les états finaux des automates orbitaux que nous étudions. La principale différence par rapport à la stabilité est que ces cibles communes ne sont pas nécessairement des portes d'entrée.

Remarquons alors que l'existence d'un couple synchronisant dans un automate émondé implique l'existence d'une orbite particulière :

Lemme 3.39. *Soit A un automate émondé tel que $\mathcal{S}_A$ ne soit pas vide. Alors, il existe une orbite O tel que :*

1. *O est non triviale et contient tous les états s des couples (a, s) de $\mathcal{S}_A$;*
2. *O est accessible depuis tout état de A et n'a pas de transition de sortie ;*

Démonstration. Soit (a, s) un élément de $\mathcal{S}_A$. Comme A est émondé, s est co-accessible. De ce fait, il peut accéder à n'importe quel état de synchronisation, et il en est de même pour les autres états de synchronisation. Donc ils appartiennent tous à la même orbite que nous nommons O.

Puisque tous les états sont co-accessibles, et que tous les états finaux ont une transition vers O, alors O est accessible depuis tous les états de A. Par conséquent, O est maximale et n'a pas de transition de sortie.

Supposons maintenant que O ne contienne qu'un seul état. Comme il s'agit d'un état de synchronisation final, ce dernier a donc une transition sur lui-même. Donc O est forcément non triviale. ■

L'existence d'un couple synchronisant implique donc que le langage reconnu est infini. Similairement au graphe déstabilisé, nous définissons alors un nouvel automate dans lequel un de ces couples est retiré :

Définition 3.40. Soient A un automate et (a, s) un élément de $\mathcal{S}_A$. La *coupure par* (a, s) *de* A est l'automate $A^{-(a,s)}$, obtenu en supprimant les transitions (f, a, s) de A pour tout état f final. Le langage droit d'un état q dans $A^{-(a,s)}$ est appelé le *langage de coupure par* (a, s) *de* q, noté $\mathrm{L}^{-(a,s)}(q)$, et forme avec ceux des autres états l'ensemble des langages de coupure par (a, s) de l'automate.

Il est alors possible d'exprimer le langage droit de chaque état uniquement à partir de langages de coupure :

Lemme 3.41. *Soient A un automate tel que $\mathcal{S}_A$ ne soit pas vide, q un état de A et (a, s) un élément de $\mathcal{S}_A$. Alors :*

1. *q est co-accessible dans $A^{-(a,s)}$ si et seulement s'il l'est dans A,*
2. $\mathrm{L}(q) = \mathrm{L}^{-(a,s)}(q) \cdot (\{\varepsilon\} \cup \{a\} \cdot \mathrm{L}(s))$
   $\mathrm{L}^{-(a,s)}(q) \cdot (\{a\} \cdot \mathrm{L}^{-(a,s)}(s))^*$

Démonstration. Remarquons d'abord que si q est co-accessible dans $A^{-(a,s)}$, alors il l'est dans A. Si maintenant q n'est pas co-accessible dans $A^{-(a,s)}$, les seules transitions supprimées par une coupure étant celles sortant des états finaux, alors il n'y a aucun chemin allant de q à un état final dans A.

D'après la Définition 3.40, le langage de coupure de q est composé de mots étiquetant des chemins acceptant, partant de q et ne contenant aucune transition de coupure. Ces mots appartiennent donc également au langage droit de q, et peuvent être prolongés par un mot du langage droit de l'état de synchronisation en utilisant une transition de synchronisation. Soit :

$$\mathrm{L}(q) = \mathrm{L}^{-(a,s)}(q) \cdot (\{\varepsilon\} \cup \{a\} \cdot \mathrm{L}(s))$$

Remarquons alors que, dans le cas de s, nous avons :

$$\begin{aligned}
\mathrm{L}(s) &= \mathrm{L}^{-(a,s)}(s) \cup \mathrm{L}^{-(a,s)}(s) \cdot \{a\} \cdot \mathrm{L}(s) \\
&= (\mathrm{L}^{-(a,s)}(s) \cdot \{a\})^* \cdot \mathrm{L}^{-(a,s)}(s) \text{ (Lemme d'Arden [3])} \\
&= \mathrm{L}^{-(a,s)}(s) \cdot (\{a\} \cdot \mathrm{L}^{-(a,s)}(s))^*
\end{aligned}$$

Ce qui permet de conclure :

$$\begin{aligned}
\mathrm{L}(q) &= \mathrm{L}^{-(a,s)}(p) \cdot (\{\varepsilon\} \cup \{a\} \cdot \mathrm{L}(s)) \\
&= \mathrm{L}^{-(a,s)}(q) \cdot (\{\varepsilon\} \cup \{a\} \cdot \mathrm{L}^{-(a,s)}(s) \cdot (\{a\} \cdot \mathrm{L}^{-(a,s)}(s))^*) \\
&= \mathrm{L}^{-(a,s)}(q) \cdot (\{a\} \cdot \mathrm{L}^{-(a,s)}(s))^*
\end{aligned}$$

■

Nous souhaitons également faire le cheminement inverse en calculant le langage de coupure par une lettre a d'un langage déterministe à partir d'une expression rationnelle déterministe le dénotant. Selon la Définition 3.40 et les résultats du Lemme 3.41, ce langage est égal au langage de départ L auquel est retiré tous les mots de la forme uav tel que u est un mot du langage de départ, soit $L \setminus (L \cdot \{a\} \cdot \Sigma^*)$.

Soient E une expression rationnelle déterministe, et a une lettre de son alphabet. Nous dénotons par $\mathrm{L}^{-a}(E)$ la coupure par a du langage dénoté par E. Puisque E est déterministe, le calcul de cette dernière consiste alors à supprimer a des Followlast, ainsi que des First si E dénote le mot vide. Nous définissons alors la coupure d'une expression déterministe et nous prouvons que le langage dénoté est celui recherché :

Définition 3.42. Soient E une expression rationnelle déterministe, émondée et en forme normale étoilée, et a une lettre de son alphabet Σ. La *coupure de E par a* est l'expression rationnelle E^{-a} calculée inductivement de la manière suivante :

— si E vaut $\emptyset$, ε ou un symbole alphabétique b, alors $E^{-a} = E$;

— si $E = F + G$, alors trois cas se présentent :

$$\begin{cases} \left(\sum\limits_{b \neq a} b \cdot \mathrm{d}_b(F^{-a})\right) + G^{-a} & \text{si } a \in \mathrm{First}(F) \wedge \mathrm{Null}(E) \wedge \neg(\mathrm{Null}(F)) \\ F^{-a} + \left(\sum\limits_{b \neq a} b \cdot \mathrm{d}_b(G^{-a})\right) & \text{si } a \in \mathrm{First}(G) \wedge \mathrm{Null}(E) \wedge \neg(\mathrm{Null}(G)) \\ F^{-a} + G^{-a} & \text{sinon} \end{cases}$$

— si $E = F \cdot G$, alors deux cas se présentent :

$$\begin{cases} F^{-a} \cdot G^{-a} & \text{si } \mathrm{Null}(G) \\ F \cdot G^{-a} & \text{sinon} \end{cases}$$

— si $E = F^*$, alors deux cas se présentent :

$$\begin{cases} \left(\sum\limits_{b \neq a} b \cdot \mathrm{d}_b(F)\right)^* & \text{si } a \in \mathrm{First}(F) \\ (F^{-a})^* & \text{sinon} \end{cases}$$

Lemme 3.43. *Soient E une expression rationnelle déterministe, émondée et en forme normale étoilée, et a une lettre de son alphabet Σ. Alors* $\mathrm{L}(E^{-a}) = \mathrm{L}^{-a}(E)$.

Démonstration. Remarquons que si un langage contient moins de deux mots, alors il est égal à sa coupure. Donc si E vaut $\emptyset$, ε ou un symbole alphabétique b, alors $\mathrm{L}^{-a}(E) = \mathrm{L}(E)$.

Si $E = F + G$, alors l'intersection de $\mathrm{First}(F)$ avec $\mathrm{First}(G)$ est vide. Donc pour tout mot u non vide, et tout mot v, si u est dans $\mathrm{L}(F)$, alors uv n'est pas dans $\mathrm{L}(G)$. Et inversement, si u est dans $\mathrm{L}(G)$, alors uv n'est pas dans $\mathrm{L}(F)$. Puisque seul ε peut appartenir aux deux langages, alors $\mathrm{L}^{-a}(E)$ est égal à $\big(\mathrm{L}^{-a}(F) \cup \mathrm{L}^{-a}(G)\big) \setminus \Big(\big(\mathrm{L}(E) \cap \{\varepsilon\}\big) \cdot \{a\} \cdot \Sigma^*\Big)$.
Un premier résultat est que si $\mathrm{Null}(E)$ n'est pas vérifiée ou que a ne fait pas partie de $\mathrm{First}(E)$, alors $\mathrm{L}^{-a}(E)$ est égal à $\mathrm{L}^{-a}(F) \cup \mathrm{L}^{-a}(G)$. Supposons maintenant que $\mathrm{Null}(E)$ est vérifiée et que a fait partie de $\mathrm{First}(E)$. Alors $\big(\mathrm{L}(E) \cap \{\varepsilon\}\big) \cdot \{a\} \cdot \Sigma^*$ est égal à $\{a\} \cdot \Sigma^*$ et deux cas se présentent.
Si a est dans $\mathrm{First}(F)$ et que $\mathrm{Null}(F)$ est vérifiée, alors $\{a\} \cdot \Sigma^*$ est inclus dans $\mathrm{L}(F) \cdot \{a\} \cdot \Sigma^*$, qui n'est pas conservé dans $\mathrm{L}^{-a}(F)$. Comme a n'est pas dans $\mathrm{First}(G)$ alors, là encore, $\mathrm{L}^{-a}(E)$ est égal à $\mathrm{L}^{-a}(F) \cup \mathrm{L}^{-a}(G)$. Nous obtenons le même résultat en appliquant le même raisonnement lorsque a est dans $\mathrm{First}(G)$ et que $\mathrm{Null}(G)$ est vérifiée.
Si maintenant a est dans $\mathrm{First}(F)$ et que $\mathrm{Null}(F)$ n'est pas vérifiée, alors il faut retirer $\{a\} \cdot \Sigma^*$ de $\mathrm{L}^{-a}(F)$. Comme a n'est pas dans $\mathrm{First}(G)$ alors, d'après le Lemme 1.14, nous obtenons que :

$$\mathrm{L}^{-a}(E) = \left(\bigcup_{b \neq a} \{b\} \cdot b^{-1}\big(\mathrm{L}^{-a}(F)\big)\right) \cup \mathrm{L}^{-a}(G)$$

Le dernier cas s'obtient par un raisonnement similaire lorsque a est dans $\mathrm{First}(G)$ et que $\mathrm{Null}(G)$ n'est pas vérifiée.

Si $E = F \cdot G$, alors l'intersection de $\mathrm{Followlast}(F)$ avec $\mathrm{First}(G)$ est vide, et si $\mathrm{Null}(F)$ est vérifiée alors l'intersection de $\mathrm{First}(F)$ avec $\mathrm{First}(G)$ l'est également. Alors pour tout w dans $\mathrm{L}(E)$, il existe un unique couple de mots (u, v) tel que $w = uv$ avec u un mot de $\mathrm{L}(F)$ et v un mot de $\mathrm{L}(G)$.
Dans un premier temps, il faut donc supprimer a de $\mathrm{Followlast}(G)$ qui est nécessairement inclus dans $\mathrm{Followlast}(E)$. Cela revient à calculer le langage intermédiaire suivant :

$$L_1 = \mathrm{L}(F) \cdot \mathrm{L}^{-a}(G)$$

Supposons alors que $\mathrm{Null}(G)$ ne soit pas vérifiée. Tous les mots de $\mathrm{L}(E)$ se terminent alors nécessairement par un mot non vide de $\mathrm{L}(G)$, et $\mathrm{L}^{-a}(E)$ est donc égal à L_1.
Supposons maintenant que $\mathrm{Null}(G)$ soit vérifiée. Remarquons que si l'ensemble $\mathrm{First}(G^{-a})$ est alors inclus dans $\mathrm{Followlast}(E)$, a n'en fait pas partie puisque ε est dans $\mathrm{L}(G)$. Il s'agit alors de supprimer également a de

Followlast(F), et de First(F) si Null(F) est vérifiée. Soit :

$$\mathrm{L}^{-a}(E) = \mathrm{L}^{-a}(F) \cdot \mathrm{L}^{-a}(G)$$

Si $E = F^*$, alors l'intersection de Followlast(F) avec First(F) est vide. Donc pour tout w dans $\mathrm{L}(E)$, il existe une unique série de mots $u_1, u_2, \cdots, u_n$ dans $\mathrm{L}(F)$ tel que $w = u_1 u_2 \cdots u_n$.
Supposons que a est dans First(F). Alors a n'est pas dans Followlast(F). Mais comme Null(E) est vérifiée, il faut retirer $\{a\} \cdot \Sigma^*$ de $\mathrm{L}(E)$, et donc de $\mathrm{L}(F)$. Or, E étant en forme normale étoilée, Null(F) n'est pas vérifiée et $\{a\} \cdot \Sigma^*$ n'est donc pas inclus dans $\mathrm{L}(F) \cdot \{a\} \cdot \Sigma^*$. En employant le Lemme 1.14, nous obtenons que :

$$\mathrm{L}^{-a}(E) = \left(\bigcup_{b \neq a} \{b\} \cdot b^{-1}(\mathrm{L}(F)) \right)^*$$

Supposons maintenant que a n'est pas dans First(F). Il suffit alors de retirer $\mathrm{L}(F) \cdot \{a\} \cdot \Sigma^*$ de $\mathrm{L}(F)$, soit $\mathrm{L}^{-a}(E) = (\mathrm{L}^{-a}(F))^*$. ■

Ce qui permet d'établir la relation suivante entre le langage reconnu par un AFD et ses langages de coupure :

Proposition 3.44 ([14])**.** *Soit A un AFD émondé et (a, s) un élément de $\mathcal{S}_A$. Alors $\mathrm{L}(A)$ est déterministe si et seulement si les langages de coupure par (a, s) de A sont déterministes. Si $\mathrm{L}(A)$ est déterministe, alors une expression rationnelle déterministe le dénotant peut être construite à partir d'expressions rationnelles déterministes dénotant les langages de coupure.*

Démonstration. Supposons que les langages de coupure par (a, s) de A sont déterministes. D'après le Lemme 3.41, le langage reconnu par A est égal à $\mathrm{L}^{-(a,s)}(i) \cdot \left(\{a\} \cdot \mathrm{L}^{-(a,s)}(s)\right)^*$, avec i l'état initial de A. Alors il existe deux expressions rationnelles déterministes $E_i^{-(a,s)}$ et $E_s^{-(a,s)}$ dénotant respectivement les langages de coupures de i et s. Comme A est déterministe, aucun état final de sa coupure n'a de transition sortante étiquetée par a. Donc a n'appartient ni à Followlast($E_i^{-(a,s)}$) ni à Followlast($E_s^{-(a,s)}$). Si i est final, alors a n'appartient pas non plus à First($E_i^{-(a,s)}$). Donc, d'après le Lemme 3.23, l'expression $E_i^{-(a,s)} \cdot (a \cdot E_s^{-(a,s)})^*$ est déterministe.

Supposons maintenant que le langage reconnu par A est déterministe. Il existe donc une expression rationnelle E le dénotant tel que E est déterministe, émondée et en forme normale étoilée. Puisque A est émondé, pour tout état q de A, il existe un mot w tel que $\mathrm{L}_A(q) = w^{-1}(\mathrm{L}(A))$. La dérivée de E par w est donc une expression rationnelle déterministe, émondée,

en forme normale étoilée et dénotant $\mathrm{L}_A(q)$. Il suffit alors de montrer par induction que le calcul de l'expression de coupure produit une expression rationnelle déterministe et en forme normale étoilée.
Les cas de base sont bien déterministes et en forme normale étoilée. Nous examinons alors les différents cas inductifs. Remarquons au préalable que Null(E) est vérifiée si et seulement si Null(E^{-a}) l'est, et que First(E^{-a}) et Followlast(E^{-a}) sont des sous-ensembles respectifs de First(E) et de Followlast(E).
Si $E = F+G$, sa coupure est dans les trois cas une somme tel que l'ensemble des First du membre gauche (resp. droit) est inclus dans celui des First de F (resp. G). Comme E est déterministe, l'intersection de l'ensemble des First du membre gauche avec celui des First du membre droit est donc vide. Par hypothèse d'induction, F^{-a} et G^{-a} sont déterministes et en forme normale étoilée. D'après les résultats de la Section 3.2.1, leurs dérivées sont également déterministes et en forme normale étoilée. Dans les trois cas, E^{-a} est donc déterministe et en forme normale étoilée.
Si $E = F \cdot G$, sa coupure est dans les deux cas une concaténation tel que l'ensemble des Followlast du membre gauche est inclus dans Followlast(F), et l'ensemble des First du membre droit est inclus dans First(G). Comme E est déterministe, l'intersection de l'ensemble des Followlast du membre gauche avec celui des First du membre droit est donc vide. Par hypothèse d'induction, F^{-a} et G^{-a} sont déterministes et en forme normale étoilée. Si Null(F^{-a}) n'est pas vérifiée, alors E^{-a} est déterministe et en forme normale étoilée. Si elle l'est, alors Null(F) l'est aussi et l'intersection de First(F) avec First(G) est vide. Comme First(F^{-a}) est inclus dans First(F), alors l'intersection de First(F^{-a}) avec First(G^{-a}) est vide.
Si $E = F^*$, sa coupure est dans les deux cas l'étoile d'une expression dont l'ensemble des First et des Followlast sont respectivement inclus dans First(F) et Followlast(F). Comme E est déterministe, l'intersection de ces deux ensembles est donc vide. De plus, comme E est déterministe et en forme normale étoilée alors F l'est aussi. Dans le premier cas, cela signifie que les dérivées de F sont déterministes et en forme normale étoilée, que leurs sommes n'incluent pas le mot vide, et donc que E^{-a} est déterministe et en forme normale étoilée. Dans le second cas, cela signifie que Null(F) n'est pas vérifiée, et donc que Null(F^{-a}) n'est pas vérifiée. Par hypothèse d'induction, F^{-a} est déterministe et en forme normale étoilée, et donc E^{-a} aussi. ■

3.2.3.3 Le BW-test et ses propriétés

Brüggemann-Klein et Wood se servent des résultats des Propositions 3.37 et 3.44 pour formuler un test consistant à couper d'abord toutes les transitions de synchronisation pour ensuite tester les automates orbitaux de l'automate obtenu. Ce test est défini à la base uniquement sur les AFD minimaux, même s'il est possible de prouver que si un AFD le valide, alors il reconnaît un langage déterministe. Cependant, nous définissons un test différent pour les besoins du Chapitre suivant. Notre test suit une logique similaire à celui décrit par le Théorème 2.28 pour caractériser les automates des positions : nous vérifions d'abord qu'un AFD satisfait la propriété orbitale, puis que ses orbites non triviales ont toutes un couple de synchronisation, pour enfin tester récursivement les automates de coupure. Toutefois, nous pourrons montrer que le test d'un seul automate de coupure suffit :

Définition 3.45. Soit A un automate. Le BW-test s'exécute de la façon suivante :

1. si A ne satisfait pas la propriété orbitale, alors le test s'arrête et échoue
2. pour toute orbite non triviale O de A :
   a) sélectionner un automate orbital B de O
   b) si $\mathcal{S}_B$ est vide, alors le test s'arrête et échoue
   c) soit un élément (a, s) de $\mathcal{S}_B$, tester récursivement $B_s^{-(a,s)}$
3. le test réussit.

Ce test peut s'appliquer à n'importe quel automate, et non seulement aux déterministes. Cela nous permettra de nous en servir dans les prochains Chapitres. Remarquons que l'automate $B_s^{-(a,s)}$ utilisé à l'appel récursif est nécessairement plus petit que A (d'au moins une transition synchronisante), et donc que ce test se termine toujours.

Nous commençons par montrer que le choix de l'élément dans $\mathcal{S}_B$ n'a pas d'importance.

Lemme 3.46. *Soient B un automate orbital, et (a, r) et (b, s) deux couples de synchronisation de B distincts. Alors $B_r^{-(a,r)}$ valide le BW-test si et seulement si $B_s^{-(b,s)}$ le valide.*

Démonstration. Prenons l'automate $B_r^{-(a,r)}$. Son ensemble de couples synchronisants est alors égal à $\mathcal{S}_B \setminus \{(a, r)\}$. De plus, selon le Lemme 3.39, il existe une orbite "synchronisante" O_S incluant tous les états des couples de synchronisation restants. Remarquons alors que certaines transitions de synchronisation de l'unique orbite de B ne sont pas nécessairement des

transitions internes de O_S, mais des transitions de sortie d'autres orbites. Soit O l'une de ces orbites. La structure interne de O ne contient ni état de synchronisation ni transition de synchronisation. Cette orbite serait alors nécessairement apparue lors de l'exécution du BW-test sur $B_s^{-(b,s)}$. Si sa structure interne serait bien la même, son ensemble de transitions de sortie ne le serait pas nécessairement si ses portes sont finales. En effet, ces transitions auraient pu être supprimées par la coupure d'un autre couple de synchronisation lors d'une exécution précédente du BW-test. Toutefois, elles sont supprimées pour tous les états finaux ou pas du tout. Donc O est transverse lors d'un test par un des couples de synchronisation si et seulement si elle l'est lors d'un test par un autre couple. L'automate orbital de O étant le même, le BW-test s'exécute de la même manière dessus.

À mesure que ces orbites sont réduites par les coupures, cela crée des orbites apparaissant dans toutes les exécutions du BW-test. Donc, à l'exception des orbites synchronisantes, toutes les autres orbites apparaissent lors de l'exécution du BW-test, peu importe le couple de synchronisation choisi. Les résultats des différents tests possibles sont donc équivalents. ■

Nous pouvons alors montrer que, dans le cas des AFD, la validation de ce test implique le déterminisme du langage :

Proposition 3.47. *Si un AFD A valide le BW-test, alors* $\mathrm{L}(A)$ *est déterministe et une expression rationnelle déterministe le dénotant peut être construite.*

Démonstration. Soit A un AFD validant le BW-test. Si A ne contient que des orbites triviales, alors $\mathrm{L}(A)$ est fini et, d'après le Théorème 3.30, est déterministe.

Supposons alors que A contienne une orbite non triviale O. Soit B l'un des automates orbitaux de O. Puisque A valide le BW-test, alors il existe un élément (a, s) dans $\mathcal{S}_B$ tel que $B_s^{-(a,s)}$ valide le BW-test. Puisque ce dernier est structurellement plus petit que A, nous déduisons par induction que son langage est déterministe.
Nous nous intéressons alors aux autres langages de coupure de B. Ce dernier étant un automate orbital, il existe donc pour tout état q de B un chemin allant de s vers q, qui est donc préservé après coupure par (a, s). D'après le Lemme 3.41, $B_s^{-(a,s)}$ est donc émondé et tous les langages de coupure par (a, s) de B sont des résiduels de celui de s. Nous concluons avec le Théorème 3.29 que tous les langages de coupure de B sont déterministes.
La Proposition 3.44 nous permet alors de conclure que le langage orbital de s dans A est déterministe. Puisqu'il s'agit d'un automate orbital, alors tous les autres langages orbitaux de O sont des résiduels de celui de s et, encore

d'après le Théorème 3.29, sont donc déterministes. Donc, tous les langages orbitaux de A sont déterministes, et ce dernier valide la propriété orbitale. D'après la Proposition 3.37, $\mathrm{L}(A)$ est déterministe.

La construction de l'expression rationnelle suit les éléments donnés dans les preuves des Propositions 3.37 et 3.44. ■

Nous examinons la complexité de l'exécution du BW-test. Le test de la propriété orbitale implique de calculer les orbites de l'automate, ce qui se fait en temps linéaire sur le nombre de transitions. Le calcul des couples de synchronisation se fait également en temps linéaire sur le nombre de transitions. Quant au nombre d'automates qui doivent être testés, celui-ci est, au pire, égal au nombre de transitions de l'automate de départ puisque ce dernier pourrait être composé d'une unique orbite non triviale avec une unique transition de synchronisation. Dans le pire des cas, ce test se fait donc en temps quadratique sur le nombre de transitions, c'est-à-dire en temps quadratique sur le nombre d'états dans le cas des AFD. Quant à l'expression rationnelle déterministe construite, Losemann *et al.* [44] ont montré que pour certains langages, les plus petites expressions rationnelles déterministes les dénotant ont une taille exponentielle par rapport à la taille de l'AFD minimal.

Théorème 3.48 ([14, 44])**.** *Soit un langage rationnel L reconnu par un AFD A. L'application du BW-test sur A se fait en temps $O(|Q_A|^2)$. Si A valide le BW-test, alors une expression rationnelle déterministe dénotant L peut être construite en temps $O(e^{|Q_A|})$.*

Exemple 3.49. Nous exécutons le BW-test sur l'AFD minimal A représenté en Figure 3.7. L'unique orbite non triviale est bien transverse. Nous sélectionnons alors l'automate orbital A_4 représenté en Figure 3.8. Les états finaux de ce dernier se synchronisant par b sur l'état 4, nous testons cette coupure, représentée en Figure 3.9. Cette dernière étant acyclique, le BW-test est validé. Il nous faut maintenant construire les expressions rationnelles dénotant les sous-langages nous permettant de construire celle dénotant $\mathrm{L}(A)$. Le langage reconnu par $A_4^{-(b,4)}$ est dénoté par $\varepsilon + a + ca$. Selon le Lemme 3.41, le langage orbital de l'état 3 est donc dénoté par $(b(\varepsilon+a+ca))^*$, et celui de l'état 2 par $a(b(\varepsilon+a+ca))^*$. Le langage orbital de leur orbite étant dénoté par d, selon le Lemme 3.36, les langages droits des états 2 et 3 sont donc respectivement dénotés par $a(b(\varepsilon+a+ca))^*d$ et $(b(\varepsilon+a+ca))^*d$. Selon les Propositions 3.37 et 3.44, nous en déduisons que $\mathrm{L}(A)$ peut être dénoté par l'expression rationnelle déterministe $bb(b(\varepsilon+a+ca))^*+aa(b(\varepsilon+a+ca))^*$. Ce qui est bien équivalent à l'expression $(aa + bb)(b(\varepsilon + (\varepsilon + c)a))^*d$ dont l'automate des positions, représenté en Figure 3.5, a servi à calculer A.

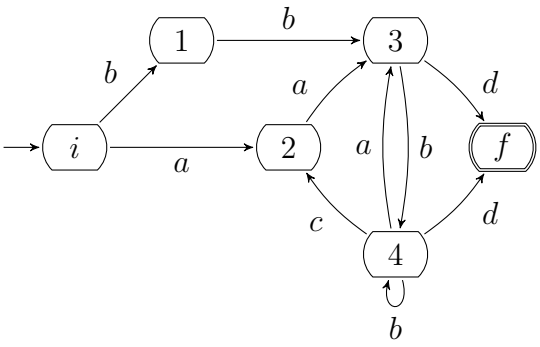


FIGURE 3.7 – L'AFD A à tester

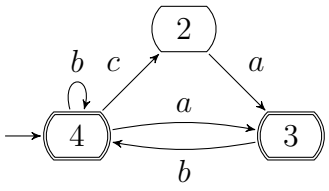


FIGURE 3.8 – L'automate orbital A_4

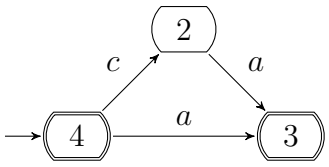


FIGURE 3.9 – La coupure $A_4^{-(b,4)}$

Pour terminer cette Section, nous souhaitons montrer que les automates des positions valident le BW-test. En effet, il existe une logique similaire entre ce test et celui décrit au Théorème 2.28 pour caractériser les automates des positions. La principale différence est que le calcul du graphe déstabilisé se fait en supprimant en une seule fois toutes les transitions de stabilité (qui sont les transitions de synchronisation).

Théorème 3.50. *Tout automate des positions valide le BW-test.*

Démonstration. Nous montrons inductivement sur les expressions rationnelles que les automates des positions valident le BW-test. Si E vaut $\emptyset$, ε ou un symbole alphabétique, alors son automate des positions est acyclique et valide donc le BW-test.
Supposons que E soit de la forme $F + G$ ou $F \cdot G$. Par hypothèse d'induction, les automates des positions de F et G valident le BW-test. Or, toutes les orbites non triviales de P_F et P_G sont présentes à l'identique dans P_E. D'après le Théorème 2.28, P_E valide la propriété orbitale et le BW-test.
Supposons alors que E soit de la forme F^*. Soit B l'un des automates orbitaux de l'unique orbite non triviale de P_E. Son ensemble de couples synchronisants n'est pas vide puisque cette orbite est stable d'après le Théorème 2.28. Par hypothèse d'induction, P_F valide le BW-test. Or, si nous

supprimons tous les couples de synchronisation de B, nous obtenons un automate équivalent à P_F privé de son état initial. En reprenant les mêmes arguments informels que pour le Lemme 3.46, les orbites non synchronisantes testées, peu importe l'ordre de retrait des couples synchronisants, sont les orbites de P_F qui sont transverses et dont les automates orbitaux valident le BW-test. Donc P_E valide le BW-test. ■

Nous en déduisons alors cette équivalence :

Théorème 3.51. *Un langage rationnel est déterministe si et seulement s'il est reconnaissable par un AFD validant le BW-test.*

Encore faut-il trouver cet AFD. Pour cela, nous montrons que si un langage est déterministe, alors son AFD minimal valide le BW-test.

3.2.3.4 Caractérisation sur l'AFD minimal

Nous détaillons les différentes étapes du BW-test lorsque celui-ci est appliqué à un AFD minimal, et donnons les éléments de preuve au fur et à mesure.

Nous commençons par montrer quelques propriétés des orbites de cet AFD minimal, héritées des orbites maximales des automates des positions déterministes :

Lemme 3.52 ([14])**.** *Soit M l'AFD minimal d'un langage déterministe. Alors :*

1. *M valide la propriété orbitale,*
2. *si M consiste en une unique orbite non triviale, alors $\mathcal{S}_M \neq \emptyset$.*

Démonstration. Soient K une orbite de M, P un automate des positions déterministe reconnaissant $\mathrm{L}(M)$, et O une orbite maximale de P tel que $K = \Omega_P(O)$. L'existence de O est assurée par le fait que Ω_P est une surjection et par la définition de la maximalité d'une orbite.

Comme nous avons pu le voir avec le Lemme 3.32, les portes de sortie de K sont les images des portes de sortie de O, et ses transitions de sortie se déduisent de celles de O. Or, d'après le Théorème 2.28, P valide la propriété orbitale. Donc K est transverse et M valide la propriété orbitale.

Supposons maintenant que K soit l'unique orbite de M et qu'elle soit non triviale. Encore d'après le Théorème 2.28, O est stable. Donc d'après le Lemme 1.44 et le Lemme 3.32, toutes les portes de sortie de K se synchronisent par les mêmes lettres sur les images des portes d'entrée de O par Φ_P. Comme K est l'unique orbite de M, alors toutes ses portes de sortie sont finales et $\mathcal{S}_M$ n'est pas vide. ■

La première propriété montre que la première partie du BW-test est validée. Remarquons toutefois que la deuxième propriété ne prouve pas que les automates orbitaux de M ont un ensemble de couples de synchronisation non vide. Pour cela, il nous faut prouver que ces automates orbitaux sont minimaux et reconnaissent des langages déterministes. Nous montrons d'abord que la minimalité est préservée grâce à la propriété orbitale et la non-ambiguïté des automates déterministes. Le fait de ne pas prendre pour hypothèse que notre automate est déterministe nous permettra de réutiliser ce résultat dans le Chapitre suivant.

Proposition 3.53. *Soit O une orbite transverse d'un automate non-ambigu A. Alors pour tout deux états p et q de O, leurs langages droits sont égaux si et seulement si leurs langages orbitaux le sont.*

Démonstration. Supposons que $\mathrm{L}^{\mathrm{orb}}(p) = \mathrm{L}^{\mathrm{orb}}(q)$. Puisque O est transverse, alors selon le Lemme 3.36, $\mathrm{L}(p) = \mathrm{L}^{\mathrm{orb}}(p)\cdot\mathrm{L}^{\mathrm{ext}}(O) = \mathrm{L}^{\mathrm{orb}}(q)\cdot\mathrm{L}^{\mathrm{ext}}(O) = \mathrm{L}(q)$.

Supposons que $\mathrm{L}^{\mathrm{orb}}(p) \neq \mathrm{L}^{\mathrm{orb}}(q)$. Soient w_i un des plus petits mots dans $\mathrm{L}^{\mathrm{orb}}(p) \triangle \mathrm{L}^{\mathrm{orb}}(q)$, et w_o un des plus petits mots dans $\mathrm{L}^{\mathrm{ext}}(O)$. Nous fixons arbitrairement que w_i est dans $\mathrm{L}^{\mathrm{orb}}(p)$. Alors w_iw_o est dans $\mathrm{L}(p)$, et il existe un chemin acceptant c composé d'un chemin interne à O étiqueté par w_i allant de p à une porte g, suivi d'un chemin acceptant étiqueté par w_o sortant immédiatement de O. Montrons que w_iw_o n'appartient pas à $\mathrm{L}(q)$:

1. Supposons que $\mathrm{L}^{\mathrm{orb}}(q)$ ne contienne aucun préfixe de w_i. Donc tout mot de $\mathrm{L}^{\mathrm{orb}}(q)$ commençant par w_i est strictement plus long que ce dernier. Comme w_o est un des plus petits mots de $\mathrm{L}^{\mathrm{ext}}(O)$, tout mot de $\mathrm{L}(q)$ commençant par w_i est strictement plus long que w_iw_o, et donc ce dernier n'est pas dans $\mathrm{L}(q)$.
2. Supposons alors que $\mathrm{L}^{\mathrm{orb}}(q)$ contienne un préfixe propre u_i de w_i, et soit v_i tel que $w_i = u_iv_i$. Puisque w_i est un des plus petits mots de $\mathrm{L}^{\mathrm{orb}}(p)\triangle\mathrm{L}^{\mathrm{orb}}(q)$, alors u_i est aussi dans $\mathrm{L}^{\mathrm{orb}}(p)$. Supposons maintenant que v_iw_o appartienne à $\mathrm{L}^{\mathrm{ext}}(O)$. Alors il existerait un chemin acceptant c' composé d'un chemin interne à O étiqueté par u_i allant de p à une porte g', suivi d'un chemin acceptant étiqueté par v_iw_o sortant immédiatement de O. Donc c et c' seraient distincts et auraient w_iw_o pour étiquette, ce qui contredirait le fait que A soit non-ambigu. Donc v_iw_o n'appartient pas à $\mathrm{L}^{\mathrm{ext}}(O)$ et w_iw_o n'appartient pas à $\mathrm{L}(q)$.

■

Puisque nous calculons l'automate orbital d'une orbite transverse d'un AFD minimal, alors cet automate est également un AFD minimal. Remarquons également une propriété annexe concernant les portes de sortie qui

nous servira juste après. En effet, un état est une porte de sortie d'une orbite si et seulement si ε est dans son langage orbital. Donc si deux états équivalents appartiennent à une même orbite transverse dans un AFD, alors l'un est une porte de sortie si et seulement si l'autre l'est.

Lemme 3.54 ([14])**.** *Soit M l'AFD minimal d'un langage déterministe. Alors les langages orbitaux de M sont déterministes,*

Démonstration. Nous ne nous intéressons qu'aux langages orbitaux des orbites non triviales puisque ceux des orbites triviales valent $\{\varepsilon\}$ et sont donc déterministes. Soient E une expression rationnelle déterministe dénotant $\mathrm{L}(M)$, P l'automate des positions déterministe de E. Soient K une orbite non triviale de M et O une orbite maximale de P tel que $K = \Omega_P(O)$.

Nous avons vu au Lemme 3.35 que les langages orbitaux des automates des positions sont tous déterministes. Nous montrons alors que les langages orbitaux des états de K sont déductibles de ceux des états de O. Soient q' un état de K, et q un de ses antécédents par Φ_P dans O dont l'existence est assuré par le Lemme 1.46.

— D'après les Lemmes 1.44 et 3.32, s'il existe (q, w, g) dans δ_P^t tel que g soit une porte de sortie de O, alors il existe $(q', w, \Phi_P(g))$ dans δ_M^t tel que $\Phi_P(g)$ est une porte de sortie de K. Le langage orbital de q est donc inclus dans celui de q'.

— D'après le Lemme 1.44, s'il existe (q', w, g') dans δ_M^t tel que g' soit une porte de sortie de K, alors il existe (q, w, g) dans δ_P^t tel que $g' = \Phi_P(g)$. Puisque O est maximale, alors g est un état de O. De plus, d'après le Lemme 3.32, il existe une porte de sortie p de O tel que $g' = \Phi_P(p)$. Comme p et g sont équivalents, d'après la remarque plus haut, comme p est une porte de sortie, alors g l'est également. Le langage orbital de q' est donc inclus dans celui de q.

Puisque tous les états de K ont un antécédent par Φ_P dans O, alors la famille des langages orbitaux des états de K est égale à celle des langages orbitaux de O. Donc les langages orbitaux de M sont déterministes. ■

Remarquons que cela prouve la réciproque de la Proposition 3.37 lorsque l'on considère le cas des AFD minimaux.

Notre automate orbital est un donc un AFD minimal consistant en une unique orbite non triviale et reconnaissant un langage déterministe. Nous pouvons donc utiliser la seconde propriété du Lemme 3.52 pour déduire que son ensemble de couples synchronisants n'est pas vide. Nous arrivons alors au test récursif sur la coupure de notre automate orbital. Afin de pouvoir utiliser notre hypothèse d'induction, il nous faut là encore montrer

que cet AFD reconnaît un langage déterministe et est minimal. La première partie est montrée par la Proposition 3.44. Il ne reste plus qu'à montrer que la coupure d'un couple synchronisant préserve et reflète la minimalité. Là encore, nous prenons pour hypothèse que notre automate est non-ambigu pour réutiliser ce résultat au Chapitre suivant.

Proposition 3.55. *Soient A un automate non-ambigu et (a, s) un élément de $\mathcal{S}_A$. Alors, pour tout deux états p et q de A, leurs langages droits sont égaux si et seulement si leurs langages de coupure par (a, s) le sont.*

Démonstration. Supposons que $\mathrm{L}^{-(a,s)}(p) = \mathrm{L}^{-(a,s)}(q)$. Selon le Lemme 3.41, $\mathrm{L}(p) = \mathrm{L}^{-(a,s)}(p) \cdot (\{\varepsilon\} \cup \{a\} \cdot \mathrm{L}(s)) = \mathrm{L}^{-(a,s)}(q) \cdot (\{\varepsilon\} \cup \{a\} \cdot \mathrm{L}(s)) = \mathrm{L}(q)$.

Supposons que $\mathrm{L}^{-(a,s)}(p) \neq \mathrm{L}^{-(a,s)}(q)$. Soit w un des plus petits mots dans $\mathrm{L}^{-(a,s)}(p) \triangle \mathrm{L}^{-(a,s)}(q)$. Nous fixons arbitrairement que w est dans $\mathrm{L}^{-(a,s)}(p)$. Donc w est dans $\mathrm{L}(p)$ et il existe un chemin acceptant c depuis p n'empruntant aucune transition de $F_A \times \{a\} \times \{s\}$ et ayant w pour étiquette. Montrons que w n'appartient pas à $\mathrm{L}(q)$:

1. Supposons que $\mathrm{L}^{-(a,s)}(q)$ ne contienne aucun préfixe de w. Comme tout mot de $\mathrm{L}(q)$ est soit un mot de $\mathrm{L}^{-(a,s)}(q)$ soit l'extension de l'un de ces mots, alors tout mot de $\mathrm{L}(q)$ commençant par w est strictement plus long que ce dernier, qui n'est donc pas dans $\mathrm{L}(q)$.
2. Supposons que w ne contienne aucune lettre a. Alors w n'est pas dans $\mathrm{L}^{-(a,s)}(q) \cdot \{a\} \cdot \mathrm{L}(s)$. Comme w n'est pas non plus dans $\mathrm{L}^{-(a,s)}(q)$, alors il n'est pas dans $\mathrm{L}^{-(a,s)}(q) \cdot (\{\varepsilon\} \cup \{a\} \cdot \mathrm{L}(s))$, c'est-à-dire $\mathrm{L}(q)$.
3. Supposons alors que $\mathrm{L}^{-(a,s)}(q)$ contienne un préfixe propre u de w, tel que w soit de la forme uav. Puisque w est un des plus petits mots de $\mathrm{L}^{-(a,s)}(p) \triangle \mathrm{L}^{-(a,s)}(q)$, alors u est aussi dans $\mathrm{L}^{-(a,s)}(p)$. Supposons maintenant que v appartienne à $\mathrm{L}(s)$. Alors il existerait un chemin acceptant c' partant de p, contenant au moins une transition de $F_A \times \{a\} \times \{s\}$, et étiqueté par w. Donc c et c' seraient distincts et auraient w pour étiquette, ce qui contredirait le fait que A soit non-ambigu. Donc v n'appartient pas à $\mathrm{L}(s)$ et w n'appartient pas à $\mathrm{L}(q)$.

■

Comme nous l'avons remarqué dans la preuve de la Proposition 3.47, l'automate $B_s^{-(a,s)}$ est émondé. Comme ce dernier est calculé à partir d'un AFD minimal, alors il s'agit également d'un AFD minimal. Par hypothèse d'induction, l'automate $B_s^{-(a,s)}$ valide le BW-test, ce qui conclut notre démonstration.

Théorème 3.56. *Un langage rationnel est déterministe si et seulement si son AFD minimal valide le BW-test.*

3.2.4 Inclusion et fermeture de la famille

Cette caractérisation sur l'AFD minimal permet de montrer simplement qu'il existe des langages rationnels qui ne sont pas déterministes. Par exemple, l'AFD représenté en Figure 3.10 est minimal et possède une unique orbite non triviale composée de deux états qui sont tous deux des portes. Cette orbite n'étant pas transverse, l'automate ne valide pas le BW-test.

Théorème 3.57. *La famille des langages déterministes est strictement incluse dans celle des rationnels.*

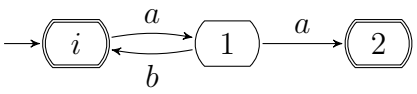
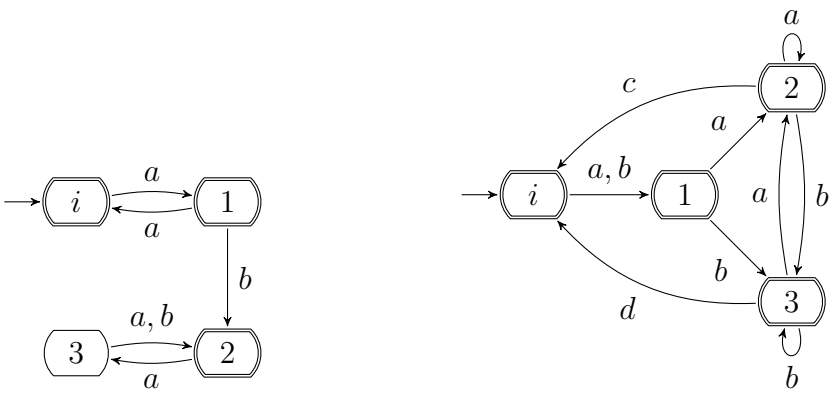


FIGURE 3.10 – L'AFD minimal du langage dénoté par $(ab)^*(\varepsilon + aa)$

Remarquons alors que si les langages dénotés par $(ab)^*$ et $\varepsilon + aa$ sont déterministes, leur concaténation ne l'est pas. La famille des langages déterministes n'est donc pas fermée par concaténation, et nous donnons d'autres exemples d'AFD minimaux montrant qu'elle ne l'est pas non plus par union, étoile de Kleene, intersection ou complémentation :

— $a^* + (a(a + b))^*$: l'AFD minimal en Figure 3.11 ne valide pas la propriété orbitale car l'orbite $\{i, 1\}$ n'est pas transverse ;

— $((a+b)(\varepsilon+ac+bd))^*$: l'AFD minimal en Figure 3.11 est une unique orbite non triviale sans couple de synchronisation ;



$a^* + (a(a + b))^*$ $\qquad$ $((a + b)(\varepsilon + ac + bd))^*$

FIGURE 3.11 – Non fermeture par union et étoile de Kleene

— $(b(\varepsilon+c))^* \cap (b+c(\varepsilon+b))(c(\varepsilon+b))^*$: l'orbite $\{1,2\}$ de l'AFD minimal en Figure 3.12 n'a pas de couple de synchronisation ;

— $\neg((aaa)^*)$: l'AFD minimal en Figure 3.12 est une unique orbite non triviale sans couple de synchronisation.

$(b(\varepsilon + c))^* \cap (b + c(\varepsilon + b))(c(\varepsilon + b))^*$ $\qquad$ $\neg((aaa)^*)$

FIGURE 3.12 – Non fermeture par intersection et complémentation

Cette famille n'est donc pas fermée par les opérateurs pour lesquels la famille des langages rationnels l'est. Elle est toutefois fermée pour quelques opérations, comme le calcul des résiduels vu au Théorème 3.29. Brüggemann-Klein et Wood [14] montrent également qu'elle est fermée par l'ajout et le retrait du mot vide. Pour cela, ils calculent une nouvelle expression rationnelle E^- avec les formules de la Table 3.1.

$$\begin{aligned}
\emptyset^- &= \varepsilon^- = \emptyset \\
a^- &= a \\
(F+G)^- &= F^- + G^- \\
(F \cdot G)^- &= \begin{cases} F^- \cdot G + G^- & \text{si } \mathrm{Null}(F \cdot G) \\ F \cdot G & \text{sinon} \end{cases} \\
F^* &= F^- \cdot F^*
\end{aligned}$$

TABLE 3.1 – Calcul de l'expression E^-

Ils montrent ensuite par induction que $\mathrm{L}(E^-) = \mathrm{L}(E) \setminus \{\varepsilon\}$, et que E est déterministe et en forme normale étoilée si et seulement si E^- l'est. Nous pouvons alors généraliser cette fermeture de la façon suivante :

Théorème 3.58. *La famille des langages déterministes est fermée par l'ajout et le retrait d'un ensemble fini de mots.*

Démonstration. Il suffit de montrer que cela vrai pour un mot non vide quelconque. Soit L un langage déterministe et w un mot. Pour ajouter ou supprimer w à L, nous calculons d'abord le résiduel de L par w qui, d'après le Théorème 3.29, est déterministe. Nous calculons ensuite le langage L_w qui

va nous servir à reconstruire l'expression attendue : si l'on souhaite ajouter w à L, alors L_w vaut $w^{-1}(L) \cup \{\varepsilon\}$; et si l'on souhaite l'en supprimer, alors L_w vaut $w^{-1}(L) \setminus \{\varepsilon\}$. Dans les deux cas, d'après les remarques plus haut, L_w est déterministe. D'après les Lemmes 1.20 et 3.23, il est alors possible de construire une expression rationnelle déterministe dénotant soit $L \cup \{w\}$ soit $L \setminus \{w\}$. ■

3.3 Conclusion

Nous avons donc pu voir que la famille des langages non-ambigus est égale à celle des rationnels tandis que celle des langages déterministes en est une partie stricte. Quelques auteurs ont présenté des extensions de la famille des langages déterministes, parfois en changeant d'approche concernant la construction de l'automate. C'est notamment le cas de Caron *et al.* [19] qui construisent un automate de manière inductive à partir des automates des sous-expressions. Cette façon de faire permet, à partir d'expressions rationnelles déterministes, de calculer un AFD reconnaissant le complémentaire d'un langage déterministe. Champarnaud *al.* [21] ont notamment montré que cette construction inductive résulte en un automate isomorphe à l'automate des positions de l'expression de départ lorsque sont employés uniquement les opérateurs $+$, $\cdot$ et *. La famille de langages de Caron *et al.*, appelée famille des langages faiblement déterministes, inclut donc strictement celle des langages déterministes et est fermée par complémentation.

Dans les deux Chapitres suivants, nous nous intéressons plutôt à des extensions basées sur des propriétés présentées par les automates des positions. Celles-ci font appel à davantage d'information que la seule prochaine lettre du mot lu en entrée, pour pouvoir décider quelle sera la prochaine position de l'expression.

Chapitre 4

Langages bloc-déterministes

Giammarresi, Montalbano et Wood [31] ont étudié une première généralisation consistant à lire au plus les k prochaines lettres du mot en entrée, pour les associer en une seule étape, et de manière unique, à une série de positions dans une expression rationnelle. De plus, ces positions doivent être concaténées en une suite ininterrompue dans l'expression, pour former un mot non vide appelé *bloc*. Un bloc a pour propriété de devoir être lu entièrement ou pas du tout, et peut donc être considéré comme une position unique dans l'expression.

Si, au lieu d'utiliser des lettres uniques comme symboles d'une expression rationnelle, nous utilisons des blocs, alors il est possible de construire des automates des positions dont les étiquettes des transitions sont des blocs. Nous étudions donc comment la notion de déterminisme peut s'étendre sur ces automates à blocs, si la famille des langages déterministes peut être étendue par l'utilisation des expressions à blocs, et la manière de caractériser cette nouvelle famille de langages.

4.1 Automates à blocs et déterminisme

L'ensemble des blocs sur un alphabet Σ est noté Γ_Σ. Un *automate à blocs* $A = (\Sigma, Q, I, F, \delta)$ (ou automate généralisé [28, 30]) est un automate dont l'ensemble des transitions δ est un sous-ensemble fini de $Q \times \Gamma_\Sigma \times Q$. Remarquons que l'ensemble δ et sa clôture transitive δ^t sont alors tous deux des sous-ensembles de $Q \times \Sigma^* \times Q$: la distinction entre ces deux ensembles permet donc de différencier une transition simple d'une séquence de transitions.

L'ensemble des blocs étiquetant les transitions de A est noté Γ_A, et l'ensemble de ceux étiquetant les transitions sortant d'un même état q de A

est noté $\Gamma_A(q)$. Si tous les blocs dans Γ_A sont de longueur inférieure ou égale à k, alors A est un automate *k-blocs*. Les automates dont les transitions sont étiquetées par des lettres sont donc des automates 1-blocs.

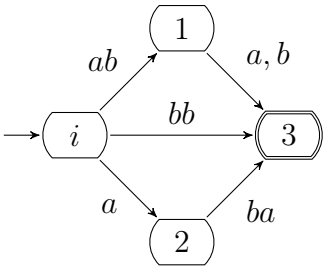


FIGURE 4.1 – Un automate 2-blocs

Pour tout automate à blocs, il existe un automate 1-blocs équivalent : ce dernier peut être construit en créant, pour toute transition (p, w, q) dont l'étiquette w est composée de plus d'une lettre, de nouveaux états intermédiaires entre p et q reliés entre eux par les lettres de w. Donc tout automate à blocs reconnaît un langage rationnel et tout langage rationnel est reconnaissable par un automate à blocs. Toutefois, dans un automate à blocs en général, et contrairement aux automates 1-blocs, le fait qu'il existe un chemin sortant d'un état p étiqueté par un mot w n'implique pas que pour tout préfixe u de w, il existe un chemin sortant de p étiqueté par u. Dans l'automate représenté sur la Figure 4.1, l'état 3 est atteint par le mot bb depuis l'état i alors qu'aucun état n'est atteint par le mot b.

Nous souhaitons maintenant généraliser le déterminisme des automates 1-blocs à n'importe quelle taille de blocs. Dans le cas des automates 1-blocs, la condition locale à chaque état que toutes les transitions sortantes soient étiquetées différemment implique qu'à chaque lettre lue sur l'entrée, il n'y a au plus qu'un seul état d'arrivée possible. Au niveau global, cela implique également qu'il existe au plus un chemin réussi pour tout mot. Cette condition n'est toutefois plus suffisante pour des tailles de blocs supérieures :

Exemple 4.1. Soit A l'automate à blocs représenté sur la Figure 4.1. Si, sur l'état i, la lettre a est lue sur l'entrée, il est possible soit de se rendre immédiatement à l'état 2, soit de rester sur l'état i pour éventuellement se rendre à l'état 1 si la lettre suivante est un b. Il faudrait alors explorer plusieurs chemins possibles pour déterminer l'appartenance du mot au langage, ce qui est une approche non déterministe. De plus, cet automate est ambigu : il existe deux chemins acceptants dont l'étiquette est aba. Ceci est dû à la possibilité de décomposer un mot en facteurs de différentes façons.

Pour pouvoir choisir la transition à emprunter parmi toutes celles sortant d'un même état, et ce uniquement à partir des étiquettes et des (au plus) k prochaines lettres sur l'entrée, il faut donc qu'au plus une de ces étiquettes soit un préfixe du mot à lire en entrée.

Définition 4.2. Un automate à blocs A est *déterministe* s'il a au plus un état initial et si pour toutes deux transitions distinctes (p, b_1, q_1) et (p, b_2, q_2), b_1 n'est pas un préfixe de b_2.

Par la suite, un automate à blocs déterministe pourra être désigné par l'abréviation ABD. Cette Définition est bien cohérente avec celle des automates 1-blocs déterministes, et le Lemme 1.25 peut alors être étendu :

Lemme 4.3. *Soient A un ABD, q un état de A et w un mot. Alors $|\delta_A^t(q, w)| \leq 1$.*

Ceci permet d'estimer la complexité du problème de l'appartenance d'un mot au langage reconnu par un automate k-blocs déterministe A. En effet, chaque lettre de Σ_A peut être associée à une valeur comprise entre 1 et $|\Sigma_A|$ par une bijection f, et chaque mot peut être associé à un entier naturel par la bijection $\mathrm{g}(w) = \sum_{1 \leq i \leq |w|} f(w[i]) \times |\Sigma_A|^{|w|-i}$. Les transitions peuvent alors être encodées dans un tableau indexé par les états et les valeurs numériques des blocs possibles. Au moment de la lecture d'un mot en entrée, chaque lettre a est lue séquentiellement en incrémentant un compteur interne (multiplication par $|\Sigma_A|$ puis ajout de $f(a)$). Ce compteur est remis à zéro dès qu'une transition valide est trouvée, et l'algorithme s'arrête si aucune transition n'est trouvée alors que k lettres ont été lues sur l'entrée après la dernière remise à zéro du compteur, ou que la fin du mot a été atteinte. Chaque lettre est donc lue au plus une seule fois et au plus un état est atteint à chaque fois, ce qui permet de conclure :

Théorème 4.4. *Soient A un ABD et w un mot. L'appartenance de w au langage reconnu par A est décidable en temps $O(|w|)$.*

Remarquons que la taille des blocs n'intervient pas dans ce modèle théorique. Celle-ci définit par contre la taille de la structure de données décrite, et intervient donc dans la complexité du problème de décision du déterminisme d'un automate k-blocs A. En effet, le plus grand langage préfixe dont la taille des mots est inférieure ou égale à k est celui contenant tous les mots de longueur k, et dont le cardinal est $|\Sigma_A|^k$. Pour chaque transition d'étiquette w, nous testons alors qu'il n'existe aucune autre transition sortant du même état et étiquetée par un préfixe de w. L'image du plus long préfixe propre de w par g est déductible de $\mathrm{g}(w)$, en décrémentant cette valeur de

1 afin que la dernière lettre soit codée entre 0 et $|\Sigma_A| - 1$, puis en calculant la partie entière de la division de la valeur obtenue par $|\Sigma_A|$. Puisqu'il peut y avoir au plus $k - 1$ préfixes propres non vides de w, nous obtenons que :

Théorème 4.5. *Le déterminisme d'un automate k-blocs A est décidable en temps $O(|Q_A| \times |\Sigma_A|^k \times k)$.*

Puisque dans un ABD, deux transitions sortant du même état ne peuvent avoir d'étiquettes dont l'une serait préfixe de l'autre, il est possible de construire un automate 1-blocs déterministe équivalent en créant de nouveaux états intermédiaires et en les reliant de sorte que les préfixes communs soient décrits par le même chemin. Pour cela, nous définissons l'opération de déblocage préfixiel d'un automate à blocs, illustrée en Figure 4.2 :

Définition 4.6. Le *débloqué préfixiel* d'un automate à blocs A est l'automate 1-blocs $(\Sigma_A, Q, I, F, \delta)$ tel que :

— $Q = \{q_\varepsilon \mid q \in Q_A\} \cup \{q_u \mid \exists (q, w, p) \in \delta_A, u \in \mathrm{Pref}(w) \setminus \{\varepsilon, w\}\}$,
— $I = \{i_\varepsilon \mid i \in I_A\}$,
— $F = \{f_\varepsilon \mid f \in F_A\}$,
— $\delta = \{(p_u, a, p_{ua}) \in Q \times \Sigma_A \times Q\} \cup \{(p_u, a, q_\varepsilon) \mid \exists (p, ua, q) \in \delta_A, a \in \Sigma_A\}$.

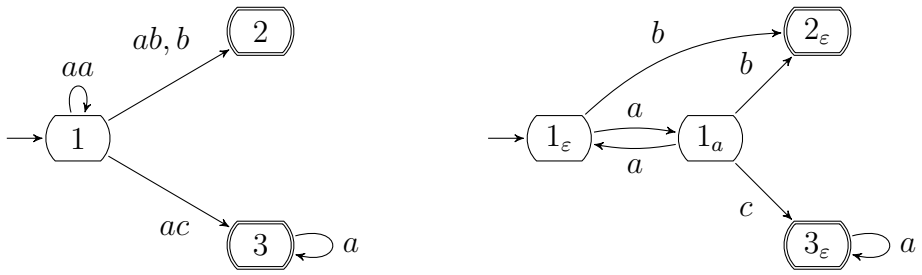


FIGURE 4.2 – Un automate à blocs et son débloqué préfixiel

Cette construction présente les propriétés suivantes :

Proposition 4.7. *Soient A un automate à blocs et D son débloqué préfixiel. Alors :*

1. *D est déterministe si et seulement si A l'est ;*
2. *il existe $(p_\varepsilon, w, q_\varepsilon)$ dans δ_D^t si et seulement s'il existe (p, w, q) dans δ_A^t ;*
3. *D est équivalent à A, et tout état q_ε de D est équivalent à q de A ;*
4. *D est émondé si et seulement si A l'est.*

Démonstration. Nous montrons d'abord que le déterminisme est préservé et reflété. Si deux transitions distinctes dans D, sortant d'un même état p_u, ont la même étiquette, alors deux cas de figure existent :

— soit elles sont de la forme (p_u, a, p_{ua}) et (p_u, a, q_ε), ce qui est équivalent à l'existence de deux transitions distinctes (p, ua, q) et $(p, uabv, r)$, avec b une lettre, dans A ;

— soit elles sont de la forme (p_u, a, q_ε) et (p_u, a, r_ε), ce qui est équivalent à l'existence de deux transitions (p, ua, q) et (p, ua, r) dans A.

Donc D n'est pas déterministe si et seulement si A ne l'est pas.

Nous nous intéressons maintenant aux chemins de D. Les chemins vides sont trivialement prouvés. Soit la transition (p, ua, q) dans δ_A avec a une lettre. Alors par définition, il existe (p_ε, u, p_u) dans δ_D^t et la transition (p_u, a, q_ε) dans δ_D. Donc $(p_\varepsilon, ua, q_\varepsilon)$ est dans δ_D^t, et ce résultat se généralise à n'importe quel chemin de A. Soit $(p_\varepsilon, ua, q_\varepsilon)$ dans δ_D^t avec a une lettre, tel que le chemin qu'il décrive ne traverse aucun état indicé par ε. Alors il existe (p_ε, u, p_u) et (p_u, a, r_ε) dans δ_D^t, et donc il existe une transition (p, ua, r) dans δ_A. Ce résultat se généralise à n'importe quel chemin de D entre deux états indicés par ε.

Il y a donc équivalence entre les chemins dans A et ceux dans D entre états indicés par ε. Étant donné qu'un état f_ε est final dans D si et seulement si f est final dans A, alors le langage droit de tout état p_ε dans D est le même que celui de p dans A. Et comme un état i_ε est initial dans D si et seulement si i est initial dans A, alors D est équivalent à A.

Pour terminer, supposons que A soit émondé. D'après les deux points précédents, tous les états de la forme p_ε de D sont accessibles et co-accessibles. Par définition, tout état de la forme p_u est accessible depuis p_ε par le mot u et est donc accessible. L'existence de cet état implique également l'existence d'une transition (p, uva, q) avec a une lettre, ce qui implique également l'existence d'un état p_{uv} dans Q_D et d'une transition $(p_{uv}, a, q_\varepsilon)$ dans δ_D. Donc p_{uv} est co-accessible et accessible depuis p_u qui est lui-même co-accessible. Donc l'automate D est émondé.
Supposons maintenant que D soit émondé. Tous les états de la forme p_ε sont donc accessibles et co-accessibles. D'après les deux points précédents, tous les états de A sont donc accessibles et co-accessibles et A est émondé. ■

La complexité de cette construction suit la même logique que celle utilisée pour justifier le Théorème 4.5, puisqu'il faut analyser chaque lettre de chaque étiquette de transitions.

Théorème 4.8. *Soit A un automate k-blocs et déterministe. Alors il est possible de construire un AFD équivalent en temps $O(|Q_A| \times |\Sigma_A|^k \times k)$.*

4.2 Propriétés des automates à blocs déterministes

La construction du débloqué préfixiel met en lumière des liens entre les ABD et les AFD. Nous étudions alors les liens pouvant exister avec l'AFD minimal du langage reconnu.

Tout d'abord, nous montrons que les langages droits d'un ABD émondé sont tous des résiduels du langage reconnu. Remarquons que l'argument employé pour prouver le Lemme 1.31, à savoir que tout état peut atteindre au plus un seul état pour tout mot, n'est pas suffisant pour étendre ce Lemme aux ABD :

Exemple 4.9. Soit A l'automate illustré en Figure 4.1. Le langage reconnu par A est $\{aba, abb\}$, et ses résiduels sont donc $\{aba, abb, bb\}$, $\{ba, bb\}$, $\{a, b\}$, $\{b\}$, $\{\varepsilon\}$ et $\emptyset$. Or, nous avons $\delta_A(i, a) = \{2\}$ alors que $\mathrm{L}(2) = \{ba\}$ n'est pas un résiduel de $\mathrm{L}(A)$.

Nous utilisons alors la construction du débloqué préfixiel et la Proposition 4.7 pour déduire le résultat suivant :

Lemme 4.10. *Soient A un ABD émondé et p un de ses états. Alors $\mathrm{L}_A(p)$ est un résiduel de $\mathrm{L}(A)$ et pour tout (p, w, q) dans δ_A^t, $\mathrm{L}_A(q) = w^{-1}(\mathrm{L}_A(p))$.*

Un ABD émondé est donc un automate résiduel. Toutefois, tous les résiduels non vides ne sont pas nécessairement présents dans sa famille de langages droits. Donc deux automates à blocs déterministes, émondés et équivalents ne sont pas nécessairement $\mathcal{L}$-équivalents. Nous pouvons cependant définir une fonction Φ associant chaque état à un état de l'AFD minimal, similaire à celle de la Définition 1.43 mais non nécessairement surjective. À partir du Lemme 4.10 et de la Proposition 4.7, nous déduisons le lien structurel suivant avec l'AFD minimal :

Corollaire 4.11. *Soient A un ABD émondé et M son AFD minimal. Si (p, w, q) est dans δ_A^t, alors $(\Phi_A(p), w, \Phi_A(q))$ est dans δ_M^t.*

Nous nous interrogeons alors sur la nécessaire présence de certains résiduels parmi les langages droits. C'est notamment le cas de ceux contenant le mot vide :

Lemme 4.12. *Soient A un ABD émondé et M son AFD minimal. Alors tout état dans F_M a un antécédent par Φ_A dans F_A.*

Démonstration. Soient f_M un état final de M et i_M son état initial, tel que (i_M, w, f_M) soit dans δ_M^t. Donc w est dans $\mathrm{L}(M)$ et dans $\mathrm{L}(A)$, et il existe un état final f_A de A tel que (i_A, w, f_A) est dans δ_A^t avec i_A l'état initial de A. D'après le Corollaire 4.11, nous avons $f_M = \Phi_A(f_A)$. ■

La fonction Φ est donc surjective lorsque sa définition est restreinte aux états finaux d'un ABD émondé. La réciproque du corollaire 4.11 est alors nécessairement vraie si q est un état final, ce qui interdit l'existence de certaines transitions :

Lemme 4.13. *Soient M un AFD minimal, q un état de M et u un mot du langage droit de q. Alors, pour tout automate à blocs A déterministe, émondé et équivalent à M, et pour tout état p dans $\Phi_A^{-1}(q)$, il n'existe pas de transition sortant de p étiquetée par un bloc w tel que u est un préfixe propre de w.*

Démonstration. Puisque p et q sont équivalents, alors il existe (p, u, f) dans δ_A^t tel que f est un état final. Ce qui implique qu'il existe un préfixe non vide u_p de u étiquetant une transition sortant de p. Donc s'il existait une transition sortant de p étiquetée par un bloc w dont u soit un préfixe propre, alors u_p serait préfixe de w et A ne serait pas déterministe. ■

En plus de cette restriction, l'ensemble des transitions se doit d'être fini. Donc si un état non final peut être effacé d'un chemin en allongeant la transition sortant de son prédécesseur dans ce chemin, un cycle non trivial entier ne peut être évité :

Lemme 4.14. *Soient A un ABD émondé, M son AFD minimal et C_M un cycle non trivial de M. Alors il existe un état dans C_M avec un antécédent par Φ_A dans Q_A, se trouvant lui-même dans un cycle non trivial.*

Démonstration. Si i_M appartient à C_M, alors i_A est un antécédent de C_M par Φ_A. De même, s'il existe un état final dans C_M, d'après le Lemme 4.12, alors cet état final a un antécédent par Φ_A dans F_A.

Supposons maintenant que C_M ne contienne ni état initial ni état final. Alors il existe :

— un mot u non vide et une porte d'entrée c_{in} de C_M tel qu'il existe un chemin étiqueté par u allant de i_M vers c_{in} sans traverser d'état de C_M,

— un mot v non vide et une porte de sortie c_{out} de C_M tel qu'il existe un chemin étiqueté par v allant de c_{out} vers un état final de M sans traverser aucun état de C_M,

— un mot w_i étiquetant un chemin interne à C_M allant de c_{in} vers c_{out}, et un mot w_o étiquetant un chemin interne à C_M allant de c_{out} vers c_{in} tel que $w_i w_o$ ne soit pas vide.

Ces éléments sont résumés en Figure 4.3.

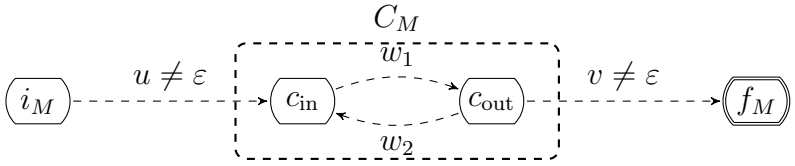


FIGURE 4.3 – Étude structurelle du cycle C_M dans la Proposition 4.14

Donc pour tout entier naturel n, le mot $uw_1(w_2w_1)^n v$ appartient au langage reconnu par M et A. Soit u_p le plus long préfixe propre de u tel qu'il existe un état q de A accessible depuis i_A par u_p, et u_s le suffixe non vide de u tel que $u = u_p u_s$. Donc pour tout entier naturel n, le mot $u_s w_1(w_2w_1)^n v$ appartient au langage droit de q dans A.
Supposons que pour tout entier naturel n, il n'existe pas de transition de A sortant de q et étiqueté par $u_s w$ avec w un préfixe de $w_1(w_2w_1)^n$. Alors pour tout entier naturel n, il existerait une transition sortant de q étiqueté par $u_s w_1(w_2w_1)^n v_n$ avec v_n un préfixe non vide de v, et l'ensemble de transitions de A serait infini. Donc il existe une transition $(q, u_s w, c)$ dans A tel que w est un préfixe de $w_1(w_2w_1)^n$ pour un entier naturel n quelconque. Ce qui, d'après le Corollaire 4.11, permet de conclure que c est un antécédent d'un état de C_M par Φ_A.

Soit S l'ensemble des antécédents de C_M par Φ_A. Cet ensemble est donc non vide et en utilisant la même logique que précédemment, tout état de S a au moins une transition sortante vers un autre état de S. Il y a donc au moins un état de S qui appartient à un cycle non trivial. ■

Or, toute présence d'un cycle non trivial implique celle d'une orbite non triviale. D'après le Corollaire 4.11, si deux états d'un ABD émondé appartiennent à la même orbite, alors leurs équivalents dans l'AFD minimal appartiennent également à la même orbite. Nous étendons donc la Définition 1.45 de la fonction Ω associant chaque orbite à une orbite de l'AFD minimal sur les automates à blocs déterministes. Cette fonction n'est pas nécessairement surjective car les orbites triviales de l'AFD minimal composées d'un état non final n'ont pas nécessairement d'antécédent. Le Lemme 4.12 implique l'existence d'antécédents pour les orbites triviales composées d'un état final, et le Lemme précédent permet de conclure pour les orbites non triviales :

Corollaire 4.15. *Soient A un ABD émondé et M son AFD minimal. Alors pour toute orbite non triviale O_M de M, il existe une orbite non triviale O_A de A tel que $O_M = \Omega_A(O_A)$.*

Enfin, nous étendons la définition d'orbite maximale aux ABD, et le Lemme 1.46 est généralisé des AFD aux ABD :

Lemme 4.16. *Soient A un ABD émondé, O_A une orbite maximale de A et O_M l'orbite de son AFD minimal M tel que $O_M = \Omega_A(O_A)$. Alors :*

- *tout état final dans O_M a un antécédent par Φ_A dans O_A,*
- *pour tout cycle non trivial C_M inclus dans O_M, il existe un état dans C_M avec un antécédent par Φ_A dans O_A.*

Démonstration. Le Lemme 4.13 permet de prouver la présence d'antécédents des états finaux de O_M dans O_A car cette dernière est maximale.

Quant aux états des cycles non triviaux, nous généralisons le Lemme 4.14 à tout état de M pouvant accéder à O_M. Soit q un état dans O_A. En considérant $\Phi_A(q)$ comme l'état initial dans M et en émondant l'automate obtenu, nous obtenons l'AFD minimal du langage droit de q dans A. Ainsi, pour chaque cycle non trivial de O_M, il existe un chemin allant de q vers un état équivalent à un état de ce cycle. Comme O_A est maximale, alors ces états sont dedans. ■

4.3 Compacité et minimalité des automates à blocs déterministes

Nous nous intéressons aux automates à blocs déterministes, compacts et émondés. Un automate 1-blocs présentant ces propriétés est minimal et unique au sens où tout autre automate équivalent les présentant également lui est isomorphe. Mais dans le cas général des ABD, ces propriétés sont nécessaires mais pas suffisantes à la minimalité car deux ABD équivalents ne sont pas nécessairement $\mathcal{L}$-équivalents.

Nous étudions donc la manière de calculer l'ensemble des automates à blocs déterministes compacts et émondés reconnaissant un langage rationnel donné par son AFD minimal et montrons, grâce aux propriétés structurelles étudiées précédemment, que pour une longueur de blocs maximale fixée, cet ensemble est fini et peut être calculé. À la suite de quoi, nous évoquons les travaux de Giammarresi et Montalbano [30] sur la minimalité en nombre d'états, dont l'un des principaux résultats est que pour un langage rationnel donné, il n'existe pas nécessairement un unique ABD minimal reconnaissant ce langage.

4.3.1 Compacité

D'après le Lemme 4.10, tout état d'un ABD émondé est équivalent à un état de son AFD minimal. Dans le cas d'un automate compact, le nombre d'états est donc inférieur ou égal à celui de l'AFD minimal. De plus, d'après le Corollaire 4.11 et le Lemme 4.13, chaque transition de l'automate à blocs est associée à un unique chemin dans l'AFD minimal qui ne traverse pas d'état final. La structure d'un automate à blocs déterministe, compact et émondé semble donc déductible de l'AFD minimal. Nous définissons alors un nouvel automate avec les mêmes ensembles d'états que ceux de l'AFD minimal, y compris initiaux et finaux, et un ensemble de transitions obtenu par clôture transitive finie et bornée sur la longueur des blocs, dont nous extrayons ensuite des sous-automates équivalents.

Définition 4.17. Soient L un langage rationnel, M son AFD minimal et k un entier strictement positif. L'*automate des k-transitions* de L est l'automate $(\Sigma_M, Q_M, I_M, F_M, \delta)$ tel que $\delta = \{(p, w, r) \in \delta_M^t \mid 1 \leq |w| \leq k \wedge \forall u \in \mathrm{Pref}(w) \setminus \{\varepsilon, w\}, u \notin \mathrm{L}_M(p)\}$.

Cette Définition permet de déduire que le langage droit de tout état d'un AFD minimal est préservé et reflété dans l'automate des k-transitions, et que ces deux automates sont équivalents.

À titre d'exemple, prenons l'AFD minimal M et son automate des 2-transitions T représentés en Figure 4.4. Remarquons que les transitions dans T correspondent bien à des chemins de M, et que les transitions $(1, aa, i)$ et $(1, ab, 1)$ n'existent pas car le mot a appartient au langage droit de 1.

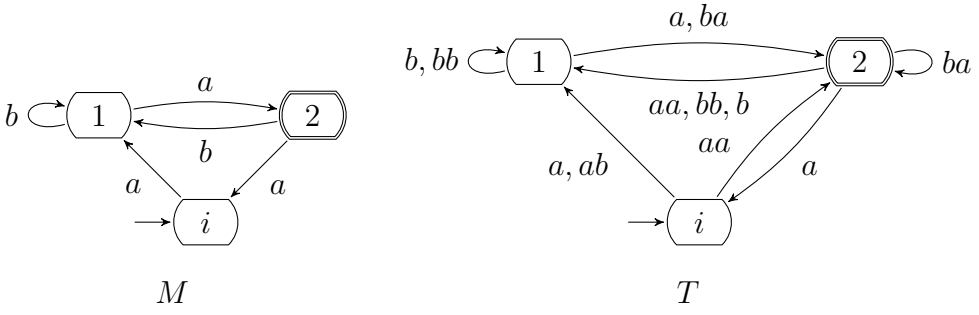


FIGURE 4.4 – Un AFD minimal M et son automate des 2-transitions T

Nous décrivons maintenant la manière d'extraire les automates recherchés comme des sous-automates de l'automate des k-transitions, le but étant que le langage reconnu et l'ensemble des langages droits des états sélectionnés soient conservés. Il faut donc garder le même état initial et, d'après le

Lemme 4.12, l'ensemble des états finaux. De plus, pour que l'automate soit déterministe et que les langages droits soient conservés, il faut sélectionner les transitions sortant d'un même état de sorte que, pour tout mot du langage droit, il existe un seul de ses préfixes étiquetant l'une d'elles.

Nous posons alors la définition suivante : un langage L_1 est *préfixe maximal* par rapport à un langage L_2 si L_1 est préfixe et si pour tout mot w dans $L_2 \setminus L_1$, le langage $L_1 \cup \{w\}$ n'est pas préfixe. Ce qui nous amène au résultat suivant :

Proposition 4.18. *Soient A un automate k-blocs et T l'automate des k-transitions d'un langage L. Alors A est déterministe, compact, émondé et reconnaît L si et seulement s'il est isomorphe à un sous-automate émondé B de T tel que :*

1. $I_B = I_T$,
2. $F_B = F_T$,
3. *pour tout état q de B, $\Gamma_B(q)$ est préfixe maximal par rapport à $\Gamma_T(q)$.*

Démonstration. Soit M l'AFD minimal de L.

Supposons que A soit isomorphe à B. D'après la Définition 4.17, pour tout état q de T, il n'existe pas deux transitions distinctes étiquetées de la même manière car T est construit à partir de M. Comme $|I_B| = |I_T| = 1$ et que pour tout état q de B, $\Gamma_B(q)$ est préfixe, alors B est déterministe. Nous montrons à présent que pour tout état q de B, le langage droit de q est le même dans B et dans T. Tout d'abord, puisque B est un sous-automate de T, alors pour tout état q de B, nous avons $\mathrm{L}_B(q) \subset \mathrm{L}_T(q)$.
L'inclusion inverse est démontrée par récurrence : pour tout entier n, pour tout état q de B, nous avons $\mathrm{L}_T(q) \cap \Sigma^n \subset \mathrm{L}_B(q)$. Le cas $n = 0$ du mot vide est immédiat car $F_B = F_T$. Supposons que la proposition soit vraie pour tout mot de longueur inférieure ou égale à un entier k. Soit q un état de B et w un mot dans $\mathrm{L}_T(q)$ de longueur $k+1$. D'après la définition de δ_T, il n'existe pas de transition sortant de q dont l'étiquette admette w comme préfixe propre. Donc il existe dans T au moins une transition sortant de q étiquetée par un préfixe non vide u de w. Comme $\Gamma_B(q)$ est préfixe maximal par rapport à $\Gamma_T(q)$, l'une de ces transitions (q, u, p) est présente dans B. Soit v un mot tel que $w = uv$. Puisque p a le même langage droit dans T et M, et que (q, u, p) appartient à δ_M^t, alors v appartient au langage droit de p dans M et T. Comme v est de longueur strictement inférieure à w, par hypothèse de récurrence, v appartient au langage droit de p dans B et donc w appartient au langage droit de q dans B.
Tous les états de B ont donc le même langage droit que dans T, et comme $I_B = I_T$, alors B reconnaît L. De plus, T étant compact, B l'est aussi.

Supposons maintenant que A soit déterministe, compact, émondé et reconnaisse L. Tout d'abord, puisque A est déterministe, émondé et reconnaît L, alors $\Phi_A(I_A) = I_M = I_T$, et d'après le Lemme 4.12, nous avons $\Phi_A(F_A) = F_M = F_T$. Comme A est compact, alors la fonction Φ_A est injective et $|Q_A| \leq |Q_T|$ et $|F_A| = |F_T|$. Soit (p, w, q) une transition de A. D'après le Corollaire 4.11, il existe donc $(\Phi_A(p), w, \Phi_A(q))$ dans δ_M^t. Comme A et T sont tous deux k-blocs, et d'après le Lemme 4.13, il existe alors aussi $(\Phi_A(p), w, \Phi_A(q))$ dans δ_T^t. Donc A est isomorphe à un sous-automate émondé B' de T tel que $I_{B'} = I_T$, $F_{B'} = F_T$. De plus, pour tout état q de B', nous avons $\mathrm{L}_{B'}(q) = \mathrm{L}_M(q) = \mathrm{L}_T(q)$. Donc, pour tout mot w dans $\mathrm{L}_T(q)$, il existe un préfixe u de w tel que u est dans $\Gamma_{B'}$, ce qui permet de conclure que $\Gamma_{B'}$ est préfixe maximal par rapport à $\Gamma_T(q)$. ■

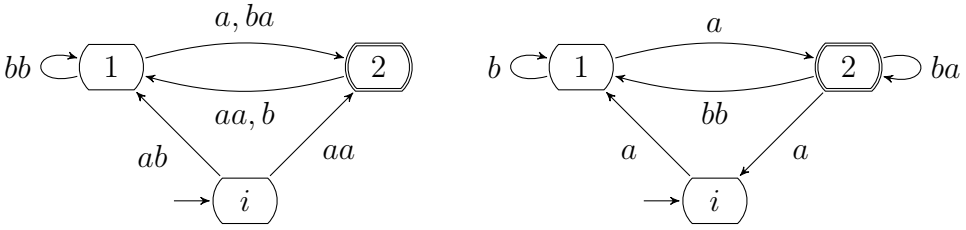


FIGURE 4.5 – Deux sous-automates de T reconnaissant $L(T)$

Tous les automates k-blocs recherchés peuvent donc être extraits de l'automate des k-transitions du langage par une sélection préfixe maximale de transitions. Afin que l'automate obtenu soit émondé, il suffit de commencer par celles sortant de l'état initial, et de continuer par celles des états atteints par les transitions précédemment sélectionnées. Les automates représentés en Figure 4.5 en sont un tel exemple. Remarquons qu'ils n'ont pas la même structure d'orbites, l'un étant composé de deux orbites dont une triviale et l'autre d'une unique orbite non triviale.

Nous en déduisons donc la finitude de l'ensemble recherché :

Théorème 4.19. *Soient L un langage rationnel et k un entier strictement positif. Alors l'ensemble des automates k-blocs, déterministes, émondés et compacts reconnaissant L est fini et calculable à partir de son AFD minimal.*

Pour terminer, nous évoquons la méthode de Giammarresi et Montalbano [30] pour compacter un ABD. Cette méthode est similaire à la minimisation des automates 1-blocs déterministes, en ce qu'elle calcule les classes d'équivalence des états par la relation d'équivalence $\sim$. Toutefois, remarquons que celle-ci n'est pas nécessairement une congruence pour les ABD,

puisque deux états peuvent être équivalents mais avec des transitions étiquetées différemment comme illustré en Figure 4.5. Le quotient d'un ABD par $\sim$, comme décrit à la Définition 1.38, est donc bien équivalent à l'automate de départ mais pas nécessairement déterministe. Aussi faut-il là aussi faire une sélection préfixe maximale de transitions sortantes, parmi celles des états d'une même classe d'équivalence.

4.3.2 Minimalité

Nous étendons la notion d'automate minimal aux automates à blocs, en ne considérant à nouveau que le nombre d'états. Giammarresi et Montalbano [30] ont étudié ces automates en utilisant une opération pour supprimer certains états tout en préservant le déterminisme, la compacité et le langage reconnu. Un ABD minimal étant nécessairement émondé et compact, il est possible d'utiliser cette opération sur l'AFD minimal d'un langage rationnel pour trouver le ou les candidats possibles.

Les propriétés étudiées précédemment nous renseignent sur les états nécessairement présents dans les candidats : les états initiaux et finaux (Proposition 4.18), et au moins un état de chaque cycle non trivial de l'AFD minimal (Lemme 4.14). Un état est alors dit *superflu* s'il n'est ni initial ni final et qu'il ne forme pas un cycle non trivial, et un automate à blocs dépourvu d'état superflu est dit *irréductible*.

Soient A un automate à blocs et q un état superflu de A. L'*élimination de l'état q dans A*, illustrée en Figure 4.6, crée un nouvel automate à blocs calculé de la manière suivante :

1. l'état q et toutes transitions y arrivant ou en sortant sont supprimés,
2. pour toutes deux transitions (p, u, q) et (q, v, r) dans δ_A, la transition (p, uv, r) est créée.

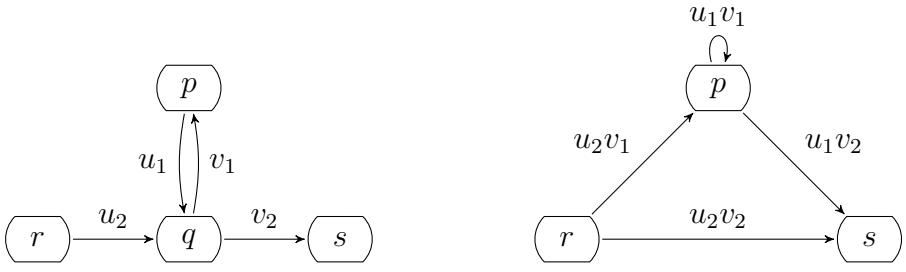


FIGURE 4.6 – Élimination de l'état q

Par analogie avec l'extraction d'automates compacts à partir d'un automate des k-transitions, l'élimination d'un état reviendrait à ne sélectionner

aucune transition y entrant. Nous montrons que cette élimination préserve les propriétés souhaitées :

Lemme 4.20. *Soient A un ABD, compact et émondé, q un état superflu de A et B l'automate à blocs obtenu par l'élimination de q dans A. Alors B est déterministe, compact, émondé et équivalent à A.*

Démonstration. L'élimination d'un état superflu préserve et reflète les étiquettes des chemins entre tout couple d'états distincts de l'état éliminé. Comme les états initiaux et finaux sont préservés, alors tous les états restants dans B ont le même langage droit que dans A, et sont accessibles et co-accessibles. Donc B est compact, émondé et équivalent à A.

De plus, A étant déterministe, les étiquettes de deux transitions distinctes sortant de q ne sont pas préfixes l'une de l'autre. Il en est de même pour tout prédécesseur direct p de q dans A. L'élimination d'état ne faisant qu'ajouter les étiquettes des transitions sortant de q comme suffixes aux étiquettes des transitions entre p et q dans A, les étiquettes de deux transitions distinctes sortant de p dans B ne peuvent pas être préfixes l'une de l'autre. Donc B est également déterministe. ■

En procédant état par état, nous aboutissons nécessairement à un automate à blocs irréductible, mais pas forcément minimal. En effet, soit M l'AFD minimal représenté en Figure 4.7. Si nous éliminons l'état 2 puis le 3, nous obtenons l'automate à blocs représenté en Figure 4.8 qui comporte trois états et est irréductible. Toutefois, si nous éliminons l'état 1, puis le 3 et enfin le 4, nous obtenons l'automate à blocs représenté à gauche en Figure 4.9 qui est irréductible et ne contient que deux états. Or, l'AFD minimal contient au moins un cycle non trivial ne contenant pas l'état initial. Tout ABD équivalent doit donc avoir au moins deux états, ce qui permet de conclure que ce dernier ABD est minimal.

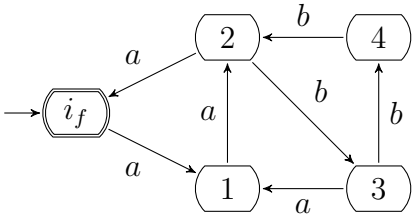


FIGURE 4.7 – L'AFD minimal M

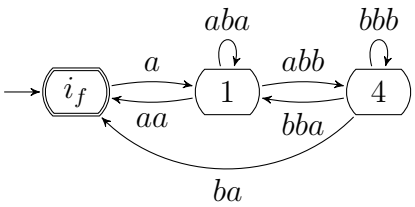


FIGURE 4.8 – Un ABD irréductible

Nous étudions alors les conditions nécessaires et suffisantes pour éliminer totalement un sous-ensemble d'états. Observons que deux états dans un au-

tomate à blocs obtenu par élimination d'un état appartienne au même cycle si et seulement s'ils appartenaient au même cycle dans l'automate à blocs de départ. Le nombre de cycles non triviaux ne variant pas, un sous-ensemble d'états superflus S peut être éliminé complètement si et seulement si aucun cycle non trivial n'est inclus dedans. De plus, l'élimination préservant et reflétant les étiquettes des chemins entre les états restants, nous obtenons le même automate à blocs peu importe l'ordre dans lequel ces états sont éliminés. Remarquons que l'élimination de j états superflus d'un automate k-blocs crée des blocs de taille au plus $j \times k$.

Afin de calculer le nombre d'états que contient un ABD minimal, il faut et il suffit donc de calculer un sous-ensemble minimum d'états de l'AFD minimal incluant l'état initial, les états finaux et un état de chaque cycle non trivial. Remarquons qu'il peut exister plusieurs sous-ensembles minimaux respectant ces critères : les ABD représentés en Figure 4.9 sont tous deux minimaux et équivalents sans être isomorphes, contrairement aux automates 1-blocs.

FIGURE 4.9 – Deux ABD minimaux équivalents

Ce qui nous permet de conclure :

Théorème 4.21 ([30]). *Il n'existe pas nécessairement un unique ABD minimal reconnaissant un langage rationnel L donné.*

4.4 La famille des langages blocs déterministes

4.4.1 Expressions rationnelles à blocs et déterminisme

Une expression rationnelle à blocs sur un alphabet Σ est une expression rationnelle dans laquelle les symboles alphabétiques sont remplacés par des blocs sur Σ. De manière analogue aux automates à blocs, si tous les blocs de E sont de longueur inférieure ou égale à k, alors E est k-blocs. Pour pouvoir distinguer les blocs comme éléments syntaxiques d'une expression à blocs, ces derniers sont écrits entre crochets, sauf pour les blocs d'une lettre.

En tant qu'éléments syntaxiques, les blocs peuvent être traités comme les symboles d'un nouvel alphabet, constituant ainsi des positions. Ceci permet de construire l'automate (à blocs) des positions d'une expression à blocs, de la même manière que précédemment, en indiçant les blocs.

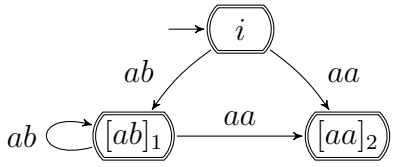


FIGURE 4.10 – L'automate des positions de $[ab]^*(\varepsilon + [aa])$

Dans le cas des expressions rationnelles à blocs, les fonctions First, Last, Follow et Followlast ne peuvent pas être définies par rapport au langage dénoté comme en Section 1.4.1 pour les expressions rationnelles à lettres. Néanmoins, elles se calculent de la même façon par les Tables données en Section 2.1 et 2.2.

Nous pouvons alors étendre naturellement la notion d'expression rationnelle déterministe aux expressions rationnelles à blocs par rapport au déterminisme de leurs automates des positions. Un langage rationnel est alors dit *k-blocs déterministe* s'il peut être dénoté par une expression rationnelle k-blocs et déterministe, et *blocs déterministe* s'il est k-blocs déterministe pour un certain k. Nous dirons également qu'un langage est *strictement* k-blocs déterministe s'il est k-blocs déterministe sans être $(k-1)$-blocs déterministe.

Par ailleurs, un automate 1-blocs déterministe étant un AFD, la famille des langages 1-blocs déterministes est la même que celle des langages déterministes. Cette dernière est même strictement incluse dans celle des blocs déterministes :

Exemple 4.22. Nous avons vu dans la preuve du Théorème 3.57 que le langage dénoté par $(ab)^*(\varepsilon + aa)$, dont l'AFD minimal est représenté en Figure 3.10 (page 76), n'est pas déterministe. Toutefois, ce langage peut être dénoté par l'expression 2-blocs $[ab]^*(\varepsilon + [aa])$ dont l'automate des positions, représenté en Figure 4.10, est bien déterministe.

Remarquons que la propriété de forme normale étoilée ne dépend que de la notion de position au sein d'une expression rationnelle linéarisée, et que cette dernière reste inchangée, peut importe la taille des blocs. Les fonctions $^\bullet$, $^\circ$ et $^\odot$ décrites en Section 2.2 sont donc applicables aux expressions rationnelles à blocs. Et à l'instar du déterminisme d'une expression rationnelle, le

déterminisme d'une expression à blocs est équivalent à l'existence d'une propriété structurelle sur l'automate des positions. Nous pouvons alors étendre le Lemme 3.22 :

Lemme 4.23. *Tout langage k-blocs déterministe peut être dénoté par une expression rationnelle k-blocs, déterministe, émondée et en forme normale étoilée.*

Nous souhaitons montrer que cette famille, comme celle des langages déterministes, contient pour chacun de ses membres l'ensemble de leurs résiduels. Nous étendons donc l'opération de dérivation aux expressions rationnelles à blocs, qui s'opère de la même façon que pour les expressions rationnelles à lettres. Seul change le cas de base où l'on dérive une expression contenant un unique bloc. Soient a et b deux lettres, et v un mot. Nous avons alors $\mathrm{d}_a([bv]) = [v]$ si $a = b$, et $\emptyset$ sinon.

Il nous faut également donner une caractérisation des expressions rationnelles à blocs étendant celle donnée au Lemme 3.23. Pour plus de lisibilité, nous introduisons l'opérateur $\cap_p$ *d'intersection préfixielle* entre deux langages L_1 et L_2 tel que $L_1 \cap_p L_2 = L_1 \cap \mathrm{Pref}(L_2) \cup L_2 \cap \mathrm{Pref}(L_1)$, représentant donc l'ensemble des mots d'un des deux langages qui sont également préfixes d'un mot de l'autre. Cet opérateur présente les propriétés suivantes :

1. il est commutatif et idempotent mais pas associatif : $\{a\} \cap_p (\{ab\} \cap_p \{ac\}) \neq (\{a\} \cap_p \{ab\}) \cap_p \{ac\}$,
2. pour tout trois langages L_1, L_2 et L_3, si L_1 est inclus dans L_2, alors $L_1 \cap_p L_3$ est inclus dans $L_2 \cap_p L_3$,
3. pour tout deux langages L_1 et L_2, $a^{-1}(L_1 \cap_p L_2) = a^{-1}(L_1) \cap_p a^{-1}(L_2)$.

Nous étendons alors le Lemme 3.23 de la façon suivante :

Lemme 4.24. *Soit E une expression rationnelle à blocs en forme normale étoilée.*

- *Si E est égale à $\emptyset$, ε ou un bloc b, alors E est déterministe ;*
- *Si $E = F + G$, alors E est déterministe si et seulement si F et G sont déterministes et* $\mathrm{First}(F) \cap_p \mathrm{First}(G) = \emptyset$ *;*
- *Si $E = FG$, alors E est déterministe si et seulement si F et G sont déterministes,* $\mathrm{Followlast}(F) \cap_p \mathrm{First}(G) = \emptyset$*, et si* $\mathrm{Null}(F)$ *est vérifiée alors* $\mathrm{First}(F) \cap_p \mathrm{First}(G) = \emptyset$ *;*
- *Si $E = F^*$, alors E est déterministe si et seulement si F est déterministe et* $\mathrm{Followlast}(F) \cap_p \mathrm{First}(F) = \emptyset$.

Ce qui nous permet d'étendre le Théorème 3.26 aux expressions à blocs :

Théorème 4.25. *Soit E une expression rationnelle à blocs sur un alphabet Σ. Si E est déterministe et en forme normale étoilée, alors pour toute lettre a de Σ, $\mathrm{d}_a(E)$ est déterministe.*

Démonstration. Tout d'abord, les remarques émises au début de la preuve du Théorème 3.26 restent valables pour les expressions à blocs. De plus, si la lettre a fait partie des blocs de $\mathrm{First}(E)$, puisque E est déterministe, alors il n'existe pas de blocs de $\mathrm{First}(E)$ ayant a pour préfixe propre, et nous retrouvons les calculs de dérivation simplifiés présentés au Lemme 3.25. Dans ce cas de figure, il suffit d'appliquer la preuve du Théorème 3.26 en remplaçant l'intersection par l'intersection préfixielle, pour arriver à la conclusion que $\mathrm{d}_a(E)$ est déterministe.

Considérons maintenant que a ne fait pas partie des blocs de $\mathrm{First}(E)$. Remarquons que dans ce cas, $\mathrm{Null}(\mathrm{d}_a(E))$ n'est pas vérifiée et $\mathrm{First}(\mathrm{d}_a(E))$ est égal à $a^{-1}(\mathrm{First}(E))$.

Si E vaut $\emptyset$, ε ou un bloc b, alors $\mathrm{d}_a(E)$ est déterministe.

Si $E = F + G$, alors $\mathrm{d}_a(E)$ est égal à $\mathrm{d}_a(F) + \mathrm{d}_a(G)$. Par hypothèse d'induction, $\mathrm{d}_a(F)$ et $\mathrm{d}_a(G)$ sont déterministes. De plus, a n'appartient ni à $\mathrm{First}(F)$ ni à $\mathrm{First}(G)$. Donc $\mathrm{First}(\mathrm{d}_a(F)) \cap_p \mathrm{First}(\mathrm{d}_a(G))$ est égal à $a^{-1}(\mathrm{First}(F)) \cap_p a^{-1}(\mathrm{First}(G))$ qui, d'après la propriété 3 de l'intersection préfixielle, vaut $a^{-1}(\mathrm{First}(F) \cap_p \mathrm{First}(G))$. Comme E est déterministe, $\mathrm{First}(F) \cap_p \mathrm{First}(G)$ est vide et $\mathrm{d}_a(E)$ est déterministe.

Si $E = F \cdot G$, alors $\mathrm{d}_a(E)$ vaut $\mathrm{d}_a(E) = \mathrm{d}_a(F) \cdot G + \mathrm{d}_a(G)$ si $\mathrm{Null}(F)$ est vérifiée, et $\mathrm{d}_a(F) \cdot G$ sinon. Puisque E est déterministe, alors G l'est aussi, et par hypothèse d'induction, $\mathrm{d}_a(F)$ et $\mathrm{d}_a(G)$ le sont également. De plus, $\mathrm{Followlast}(F) \cap_p \mathrm{First}(G)$ étant vide, alors il en est de même pour $\mathrm{Followlast}(\mathrm{d}_a(F)) \cap_p \mathrm{First}(G)$.
Supposons alors que $\mathrm{Null}(F)$ ne soit pas vérifiée. Alors a n'appartient pas à $\mathrm{First}(F)$. Donc $\mathrm{Null}(\mathrm{d}_a(F))$ n'est pas vérifiée et $\mathrm{d}_a(E)$ est déterministe.
Supposons maintenant que $\mathrm{Null}(F)$ soit vérifiée. Alors a n'appartient ni à $\mathrm{d}_a(F)$ ni à $\mathrm{d}_a(G)$. Donc $\mathrm{Null}(\mathrm{d}_a(F))$ n'est pas vérifiée et $\mathrm{First}(\mathrm{d}_a(F) \cdot G) \cap_p \mathrm{First}(\mathrm{d}_a(G))$ est égal à $\mathrm{First}(\mathrm{d}_a(F)) \cap_p \mathrm{First}(\mathrm{d}_a(G))$. Par la même logique que pour le cas $E = F + G$, nous en déduisons que $\mathrm{First}(\mathrm{d}_a(F) \cdot G) \cap_p \mathrm{First}(\mathrm{d}_a(G))$ vaut $a^{-1}(\mathrm{First}(F) \cap_p \mathrm{First}(G))$ qui est vide, et que $\mathrm{d}_a(E)$ est déterministe.

Enfin, si $E = F^*$, alors $\mathrm{d}_a(E) = \mathrm{d}_a(F) \cdot F^*$. Par hypothèse d'induction, $\mathrm{d}_a(F)$ est déterministe. De plus, comme a n'appartient pas à $\mathrm{First}(E)$ qui est égal à $\mathrm{First}(F)$, alors $\mathrm{Null}(\mathrm{d}_a(F))$ n'est pas vérifiée. Comme E est en forme normale étoilée, alors $\mathrm{Followlast}(F) \cap \mathrm{First}(F^*)$ est vide. Donc $\mathrm{Followlast}(\mathrm{d}_a(F)) \cap \mathrm{First}(F^*)$ l'est aussi, et $\mathrm{d}_a(E)$ est déterministe. ■

Pour terminer, remarquons que la dérivation n'augmente pas la taille des blocs, et donc que la dérivée d'une expression k-blocs est également k-blocs. Le Lemme 3.27 s'appliquant également aux expressions rationnelles à blocs, nous pouvons conclure que :

Théorème 4.26. *Si un langage est k-blocs déterministe, alors tous ses résiduels le sont.*

4.4.2 Extension du BW-test

Nous nous intéressons maintenant à la manière de montrer le k-blocs déterminisme d'un langage, et ce en réutilisant le BW-test. Les notions de transversalité et de couple synchronisant étant indépendantes du type d'étiquette (lettres, blocs, ...) des transitions, ce test est purement structurel et peut être appliqué directement aux automates à blocs. Il nous faut alors montrer que si un automate k-blocs et déterministe valide ce test, nous pouvons en déduire une expressions rationnelle k-blocs et déterministe.

Pour ce faire, nous travaillons sur un automate intermédiaire faisant la jonction entre un ABD et un AFD. Soient A un automate, B un automate à blocs et φ une bijection entre Σ_A et Γ_B. Nous disons que B est une *image alphabétique de A par φ* , noté $\varphi(A)$, si $Q_B = Q_A$, $I_B = I_A$, $F_B = F_A$ et $\delta_B = \{(p, \varphi(a), q) \mid (p, a, q) \in \delta_A\}$. Cette image préserve et reflète la finalité des états, les équivalences d'étiquettes des transitions arrivant dans un même état, et donc les propriétés structurelles tel que l'homogénéité, la forte stabilité et la forte transversalité. L'image alphabétique d'un automate est donc un automate des positions (respectivement valide le BW-test) si et seulement si l'automate de départ est un automate des positions (respectivement valide le BW-test). Ce qui nous permet d'étendre le Théorème 3.51 aux langages k-blocs déterministes :

Théorème 4.27. *Un langage est k-blocs déterministe si et seulement s'il est reconnaissable par un automate k-blocs, déterministe et validant le BW-test.*

Démonstration. Soit L un langage rationnel.

Si L est k-blocs déterministe, alors il est reconnu par un automate des positions k-blocs et déterministe. Le Théorème 3.50 permet de conclure.

Supposons maintenant que L soit reconnaissable par un automate à blocs B tel que B soit k-blocs, déterministe et valide le BW-test. Soient $\Pi = \{[b] \mid b \in \Gamma_B\}$ l'alphabet des blocs de B (considéré ici comme des symboles unitaire), φ la bijection entre Π et Γ_B associant à tout symbole $[b]$ de Π le bloc b de Γ_B, et A un automate tel que $B = \varphi(A)$. Comme φ est

une bijection entre Π et Γ_B et que B est déterministe, alors A est également déterministe. De plus, d'après la remarque plus haut, A valide également le BW-test. Donc d'après le Théorème 3.51, $\mathrm{L}(A)$ est déterministe et il existe un automate des positions déterministe P reconnaissant $\mathrm{L}(A)$. En appliquant alors φ^{-1}, nous obtenons un automate k-blocs G tel que $G = \varphi(P)$. Comme P est un automate des positions, G l'est également.
Nous étudions alors le langage reconnu par G. La bijection φ est étendue en un morphisme de Π^* vers Γ_B^* tel que $\varphi(\varepsilon) = \varepsilon$, et pour tout symbole $[b]$ de Π et tout mot w dans Π^*, nous avons $\varphi([b] \cdot w) = \varphi([b]) \cdot \varphi(w)$. Nous en déduisons que $\mathrm{L}(G) = \varphi(\mathrm{L}(P)) = \varphi(\mathrm{L}(A)) = \mathrm{L}(B) = L$.
Il reste alors à montrer que G est déterministe. Ce dernier étant un automate des positions, il n'a donc qu'un seul état initial. Soient $(p, \varphi(a), q)$ et $(p, \varphi(b), r)$ deux transitions distinctes de G. Par définition d'une image alphabétique, (p, a, q) et (p, b, r) sont deux transitions distinctes de P. Puisque P et A sont deux AFD équivalents, alors il existe dans A deux transitions distinctes (p', a, q') et (p', b, r') tel que p', q' et r' sont trois états de A respectivement équivalents aux états p, q et r de P. Encore par définition d'une image alphabétique, $(p', \varphi(a), q')$ et $(p', \varphi(b), r')$ sont deux transitions distinctes de B. Comme B est déterministe, alors $\varphi(a)$ n'est pas un préfixe de $\varphi(b)$ et réciproquement. Donc G est un automate des positions reconnaissant L qui est k-blocs et déterministe, et L est bien k-blocs déterministe. ■

Ce Théorème implique donc qu'il est possible de construire une expression rationnelle k-blocs et déterministe dénotant le langage reconnu par un automate k-blocs et déterministe validant le BW-test, et ce en procédant de la même manière que pour les langages déterministes. Nous ne pouvons toutefois encore rien conclure quant au k-blocs déterminisme du langage si le test venait à échouer car nous n'avons à ce stade pas d'automate canonique à tester comme l'AFD minimal dans le cas des langages déterministes.

4.4.3 Anciens résultats

Nous nous intéressons maintenant à deux problèmes précédemment étudiés, mais dont nous avons invalidé les preuves :

1. l'appartenance d'un langage à la famille des langages blocs déterministes est-elle décidable ?
2. s'il existe des langages 1-blocs déterministes et des langages strictement 2-blocs déterministes, existe t'il une hiérarchie infinie de familles de langages strictement k-blocs déterministes ?

Nous présentons donc les anciens résultats et nos contre-exemples.

Giammarresi *et al.* ont déclaré que l'opération d'élimination d'états (page 91), appliquée sur l'AFD minimal d'un langage, est nécessaire et suffisante pour décider du blocs déterminisme de ce langage :

Conjecture 1 ([31, 36])**.** *Soit M l'AFD minimal d'un langage k-blocs déterministe. Alors il est possible d'obtenir un automate k-blocs, déterministe et validant le BW-test par éliminations successives des états de M.*

Han et Wood se basent ensuite sur cette affirmation pour démontrer le résultat suivant :

Théorème 4.28 ([36])**.** *Il existe une hiérarchie infinie de familles de langages strictement k-blocs déterministes.*

Démonstration. Nous avons pu voir que la famille des langages déterministes est la même que celle des langages 1-blocs déterministes. Il reste alors à vérifier qu'il existe des langages strictement k-blocs déterministes pour tout entier k strictement supérieur à 1.

Soient k un entier supérieur ou égal à 2 et L_k le langage dénoté par l'expression $(a^k)^*(a^{k-1}bb + ba)b^*$, dont l'AFD minimal M_k est représenté en Figure 4.11. Ce langage pouvant être dénoté par l'expression $[a^k]^*$ $([a^{k-1}b]b + ba)b^*$ qui est k-blocs et déterministe, il est donc k-blocs déterministe. De plus, M_k contient une orbite non triviale et non transverse composée des états q_1 à q_k. Comme l'état q_k n'est pas superflu, d'après la Conjecture 1, il serait nécessaire d'éliminer les états q_1 à q_{k-1} de M_k afin d'obtenir l'automate N_k, représenté en Figure 4.11, qui est k-blocs, déterministe et valide le BW-test. Le langage L_k serait donc strictement k-blocs déterministe. ■

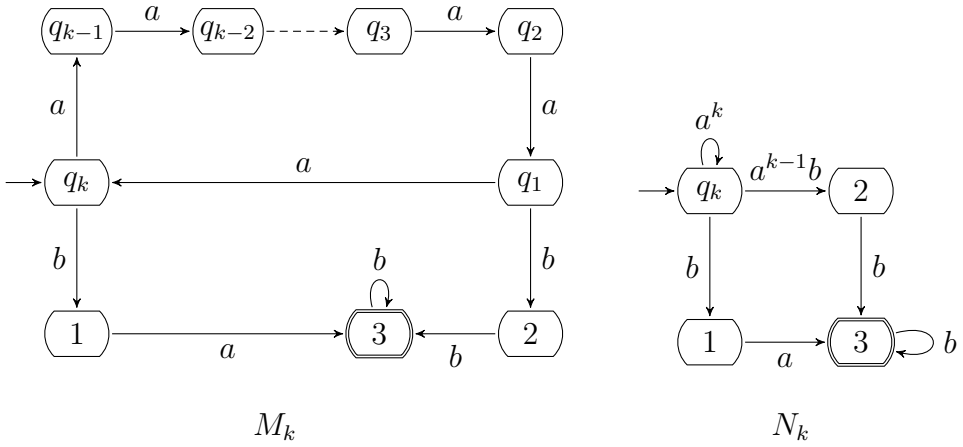


FIGURE 4.11 – Illustration de la preuve du Théorème 4.28

Nous donnons maintenant un contre-exemple à la Conjecture 1. L'AFD minimal C, représenté en Figure 4.12, ne valide pas le BW-test et n'a aucun état superflu. Selon la Conjecture 1, le langage reconnu ne serait donc pas blocs déterministe.
Remarquons alors que C est le décalé par a de l'AFD minimal A, représenté en Figure 3.10 (page 76), et reconnaît donc le résiduel par a du langage reconnu par A. Or, nous avons vu à l'Exemple 4.22 que le langage reconnu par A, dénoté par $[ab]^*(\varepsilon + [aa])$, est 2-blocs déterministe. Donc, selon le Théorème 4.26, le langage reconnu par C est bien 2-blocs déterministe.
Il est possible d'aboutir à la même conclusion en utilisant l'automate des 2-transitions de L(C) pour extraire l'automate représenté à droite en Figure 4.12, qui est 2-blocs, déterministe et valide le BW-test : il suffit de ne pas inclure la transition $(1, a, 2)$ dans la sélection préfixe maximale.
Dans les deux cas, nous pouvons en déduire que le langage reconnu par C peut être dénoté par l'expression $b[ab]^*(\varepsilon + [aa]) + a$.

L'AFD minimal C Un ABD équivalent à C

FIGURE 4.12 – Contre-exemple à la Conjecture 1

Cela montre clairement que l'opération d'élimination d'états n'est pas suffisante pour décider si un langage est blocs déterministe. En fait, elle n'est pas non plus suffisante pour décider si un langage est k-blocs déterministe, invalidant la preuve donnée par Han et Wood pour la démonstration de leur Théorème sur la hiérarchie infinie :

Proposition 4.29. *Pour tout entier k supérieur ou égal à 2, le langage L_k est 2-blocs déterministe.*

Démonstration. L'état initial q_k de l'AFD minimal M_k représenté en Figure 4.11 est non final. Nous pouvons donc créer les transitions (q_1, aa, q_{k-1}) et $(q_1, ab, 1)$ dans l'automate des 2-transitions de L_k, puis en extraire l'automate à blocs représenté en Figure 4.13 en ne sélectionnant pas la transition (q_1, a, q_k). D'après le Théorème 4.27, L_k peut donc être dénoté par l'expression 2-blocs déterministe $(a^{k-1}([aa]a^{k-2})^*([ab]a + bb) + ba)b^*$. ■

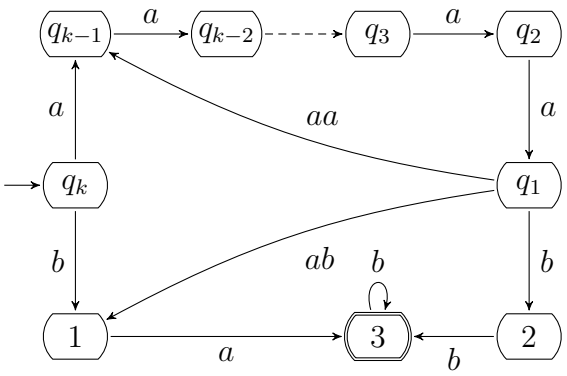


FIGURE 4.13 – Un automate 2-blocs déterministe équivalent à M_k

Dans la suite de ce Chapitre, nous présentons nos résultats concernant la décidabilité du k-blocs déterminisme (mais pas celle du blocs déterminisme) et la hiérarchie.

4.5 La décidabilité du k-blocs déterminisme

Pour décider si un langage rationnel est k-blocs déterministe, il faut tester si ce langage est j-blocs déterministe pour un entier j compris entre 1 et k. Selon le Théorème 4.27, cela revient à trouver un ABD reconnaissant le langage, qui soit j-blocs et valide le BW-test. Rappelons que dans le cas des langages déterministes, le Théorème 3.56 précise qu'il est nécessaire et suffisant que l'AFD minimal valide le BW-test. Or, pour les AFD émondés, la minimalité n'est qu'une conséquence de la compacité qui, d'après le Théorème 4.19, nous donne un ensemble fini d'ABD à tester. Il nous faut donc raffiner le Théorème 4.27 en incluant la compacité dans les conditions nécessaires et suffisantes.

Comme il a été mentionné à la fin de la Section 4.3.1, Giammarresi et Montalbano [30] décrivent une méthode de compaction préservant le déterminisme et la taille maximale des blocs en fusionnant les états équivalents puis en sélectionnant un sous-ensemble préfixe maximal de transitions. Cependant, cette méthode ne préserve pas nécessairement le BW-test.

Exemple 4.30. Soit A l'automate 2-blocs déterministe représenté en Figure 4.14. Cet automate valide le BW-test, mais n'est pas compact puisque les états X, Y et Z sont équivalents. Si on applique la méthode proposée par Giammarresi et Montalbano, nous avons deux possibilités.

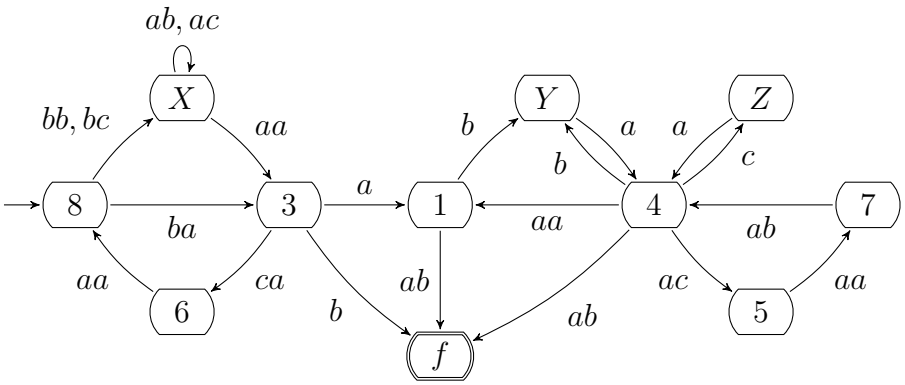


FIGURE 4.14 – L'automate A

Si nous conservons l'ensemble préfixe maximal des transitions de la classe de X composé de celle vers elle-même et de celle vers la classe de 3, nous obtenons l'automate représenté en Figure 4.15. Remarquons que les états $\{3\}$ et $\{1\}$ sont des portes de sortie de la même orbite mais n'ont pas une transition étiquetée de la même manière vers $\{f\}$.

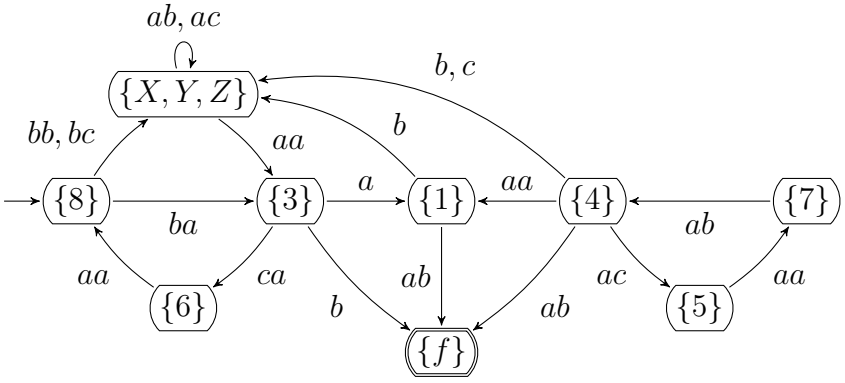


FIGURE 4.15 – Une compaction de A

Si nous conservons l'ensemble préfixe maximal des transitions de la classe de X composé de celle vers la classe de 4, nous obtenons l'automate représenté en Figure 4.16. Remarquons que les états $\{8\}$ et $\{3\}$ sont des portes de sortie de la même orbite mais n'ont pas les mêmes cibles pour leurs transitions de sortie.

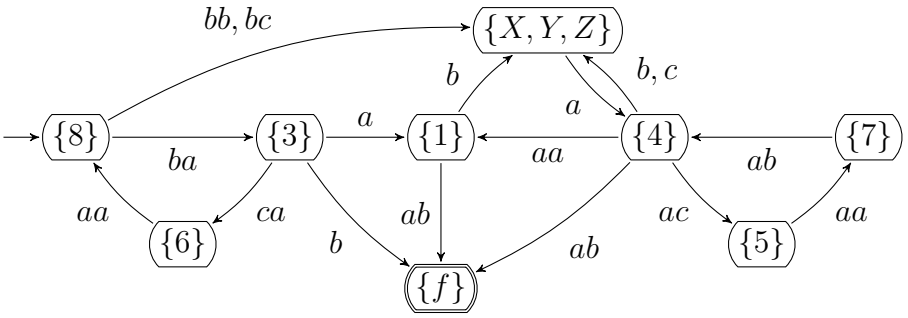


FIGURE 4.16 – L'autre compaction de A

Dans les deux cas, le BW-test n'est donc pas préservé.

Nous décrivons dans la suite de cette Section notre méthode pour compacter un automate à blocs tout en préservant la taille maximale des blocs, le déterminisme et la validité du BW-test. Remarquons que deux états équivalents peuvent aussi bien appartenir à deux orbites distinctes qu'à la même orbite. Par rapport à ce dernier cas, nous dirons d'une orbite qu'elle est *compacte* si elle ne contient aucun état équivalent. L'idée sous-jacente de notre procédure suit la Définition du BW-test :

- Compacter chaque orbite non compacte :
  1. calculer un de ses automates orbitaux
  2. retirer ses transitions de synchronisation
  3. calculer récursivement un automate à blocs compact et équivalent à celui obtenu précédemment
  4. replacer les transitions de synchronisation précédemment retirées
  5. substituer l'ancienne orbite non compacte par le nouvel automate
- Sélectionner un sous-ensemble suffisant d'orbites compactes pour les relier entre elles.

Nous serons donc amenés au cours des étapes 1 et 2 à retirer des transitions sortantes aux états finaux, ce qui préserve le déterminisme ; mais également à en rajouter au cours des étapes 4 et 5. Afin de prouver que ces étapes préservent bien le déterminisme, nous définissons *l'ensemble des étiquettes finales* d'un automate à blocs A, noté $\mathcal{F}_A$, comme étant l'ensemble des étiquettes des transitions sortant des états finaux de A, et nous montrerons que cet ensemble ou un de ses sous-ensembles est préservé. Cette notion est équivalente à l'ensemble des Followlast d'une expression, auquel il faudrait ajouter les First si le mot vide appartenait au langage dénoté.

Nous commençons par étudier la partie finale de cette procédure, consistant à compacter un automate composé uniquement d'orbites compactes. Vient ensuite la compaction des orbites elles-mêmes, et enfin la preuve globale de la procédure. Ces étapes sont illustrées dans leur déroulement par l'exemple de l'automate donné en Figure 4.14, dont la compaction entière est résumée à l'Exemple 4.44.

4.5.1 L'élimination des orbites inutiles

Plusieurs orbites pouvant avoir la même Ω-image, il s'agit ici de sélectionner un représentant de chaque Ω-image présente dans l'automate à blocs, puis de les relier entre elles pour en obtenir un autre, équivalent à l'original. Par ailleurs, si toutes les orbites sélectionnées sont compactes, alors l'automate à blocs obtenu l'est également.

Les représentants choisis ici sont des orbites maximales car ces dernières n'ont aucune transition de sortie vers des états pouvant être équivalents à ceux qu'elles contiennent. Ces orbites maximales doivent alors être relier entre elles en conservant les étiquettes de leurs transitions de sortie, afin de préserver à la fois le déterminisme et la taille maximale des blocs. Toutefois, certaines cibles de ces transitions de sortie peuvent être des états n'ayant aucun équivalent parmi les états des orbites sélectionnées.

Nous ajoutons donc ces états manquants, sous la forme d'orbites triviales, pour servir de jonction entre les orbites. Nous disons que nous complétons une orbite O avec des états de son Ω-image, appartenant dans le cas présent à l'ensemble $\Theta(O) = \Omega(O) \setminus \Phi(O)$.

4.5.1.1 Complétion d'orbites maximales

Pour servir de jonction, ces états manquants doivent avoir des transitions allant vers les états de l'orbite à compléter. Leurs images par Φ appartenant à la même orbite de l'AFD minimal correspondant, leurs langages droits sont donc des résiduels de ceux des états de l'orbite à compléter.

Nous calculons donc ces transitions directement à partir de l'automate à blocs, à la manière du calcul du décalé d'un automate. Toutefois, nous avons vu que dans le cas des automates à blocs, le fait qu'un mot appartienne au langage droit d'un état p n'implique pas que tous ses préfixes étiquettent un chemin partant de p : ce dernier pourrait se terminer par la lecture partielle de l'étiquette d'une transition. Nous utilisons alors une fonction permettant de compléter les étiquettes de ces transitions vers leurs états d'arrivées. Par mesure de simplicité, cette fonction est définie uniquement sur les automates à blocs déterministes.

Pour un ABD A, la *fonction de cheminement* χ_A de $Q_A \times \Sigma_A^*$ vers $2^{(\Sigma_A^* \times Q_A)}$ est définie par $\chi_A(p,u) = \{(v,q) \mid (p,uv,q) \in \delta_A^t \wedge \forall w \in \mathrm{Pref}(v) \setminus \{v\}, \delta_A^t(p,uw) = \emptyset\}$. Nous en déduisons les propriétés suivantes :

Lemme 4.31. *Soient A un ABD, M son AFD minimal et p un état de A. Alors :*

1. *$\chi_A(p,w) = \{(\varepsilon,q)\}$ si et seulement si (p,w,q) appartient à δ_A^t,*
2. *si $(\Phi_A(p),w,s)$ appartient à δ_M^t, alors $\mathrm{L}_M(s) = \bigcup_{(u,r)\in\chi_A(p,w)} u \cdot \mathrm{L}_A(r)$.*

Exemple 4.32. Soit A' l'automate à blocs en Figure 4.14 : nous avons $\chi(8,\varepsilon) = \{(\varepsilon,8)\}$, $\chi(8,b) = \{(a,3),\ (b,X),\ (c,X)\}$, $\chi(8,bb) = \{(\varepsilon,X)\}$, $\chi(8,bba) = \{(a,3),\ (b,X),\ (c,X)\}$ et $\chi(8,bbaa) = \{(\varepsilon,3)\}$.

Soient M l'AFD minimal de A, O une orbite de A et s un état de $\Theta(O)$. Une *complétion de O avec s* consiste, à partir d'un état o de O et d'un mot w tel que $(\Phi_A(o),w,s)$ est dans δ_M^t, à construire un automate à blocs $B = (\Sigma_A, Q, I_A, F_A, \delta)$ avec $Q = Q_A \cup \{s_A\}$ et $\delta = \delta_A \cup \{(s_A,u,r) \mid (u,r) \in \chi(o,w)\}$. Remarquons que le nouvel état s_A forme une orbite triviale.

Par ailleurs, différents choix sont possibles pour l'état o et le mot w, ce qui pourrait créer différents ensembles de transitions sortantes du nouvel état. Toutefois, toutes ces possibilités préservent les propriétés suivantes :

Proposition 4.33. *Soient A un ABD k-blocs et M l'AFD minimal de A. Soient O une orbite maximale de A et q un état de M appartenant à $\Phi(Q_A) \cap \Theta(O)$. Soit B l'automate à blocs obtenu par la complétion de O par q. Alors :*

1. *B est déterministe*
2. *$\mathcal{F}_B = \mathcal{F}_A$*
3. *B est $\mathcal{L}$-équivalent à A*
4. *B est k-blocs*
5. *si A valide le BW-test, alors B aussi.*

Démonstration. Soient p un état de O et w un mot tel que $(\Phi(p),w,q)$ appartient à δ_M^t. Soit q_A l'état ajouté pour obtenir B.

(1) : Comme q appartient à $\Theta(O)$ et que O est maximale, alors il n'existe aucun état accessible depuis O qui puisse être équivalent à q. Par définition, il existe donc un unique état s dans A et un préfixe non vide w_s de w tel que pour tout couple (u,q) dans $\chi_A(p,w)$, u est un mot non vide et la transition $(s,w_s u,q)$ est dans δ_A. Comme A est déterministe, pour tout deux couples distincts (u_1,q_1) et (u_2,q_2) dans $\chi_A(p,w)$, u_1 et u_2 ne sont pas préfixes entre eux. Donc B est déterministe.

(2) : Comme $F_B = F_A$ et $\delta_A = \delta_B \setminus (\{q_A\} \times \Sigma^* \times Q_A)$, alors $\mathcal{F}_B = \mathcal{F}_A$.

(3) : Par construction, q_A est non final ce qui permet de conclure d'après le Lemme 4.31 que q_A est équivalent à q. Tous les autres langages droits sont préservés, tout comme le langage reconnu par l'automate. Puisque q appartient à $\Phi(Q_A)$, alors A et B sont $\mathcal{L}$-équivalents.

(4) : Comme A est k-blocs, alors pour tout (u, q) dans $\chi_A(p, w)$, la longueur de u est nécessairement strictement inférieure à k.

(5) : L'automate B est formé de l'automate A auquel a été ajouté une orbite triviale. Donc si A valide le BW-test, B le valide également. ■

Exemple 4.34. Soient A' l'ABD représenté en Figure 4.17 et M son AFD minimal représenté en Figure 4.18. L'orbite $O = \{1, 4, 5, 7, Z\}$ de A' est maximale, $\Theta(O) = \{3, 6, 8\} = \Theta(O) \cap \Phi(Q_{A'})$. Les états 3, 6 et 8 de A' sont respectivement équivalents aux états 3, 6 et 8 de M. L'ABD représenté en Figure 4.19 est l'une des complétions possibles de A' avec 3, 6 et 8.

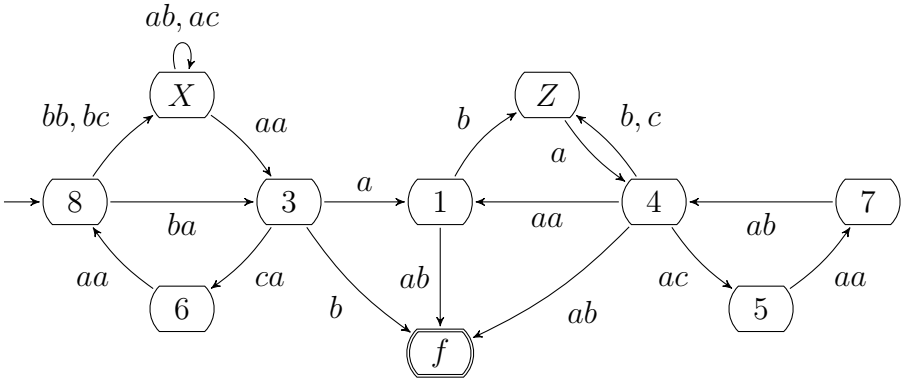


FIGURE 4.17 – L'automate A'

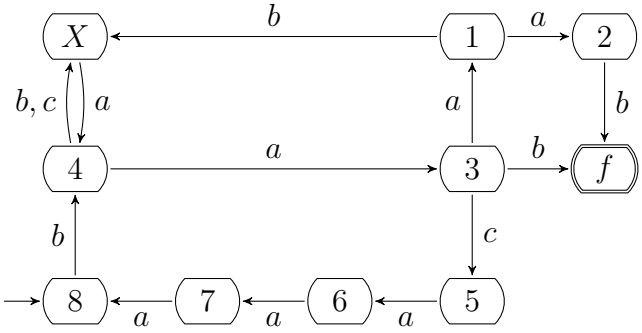


FIGURE 4.18 – L'AFD minimal M de A'

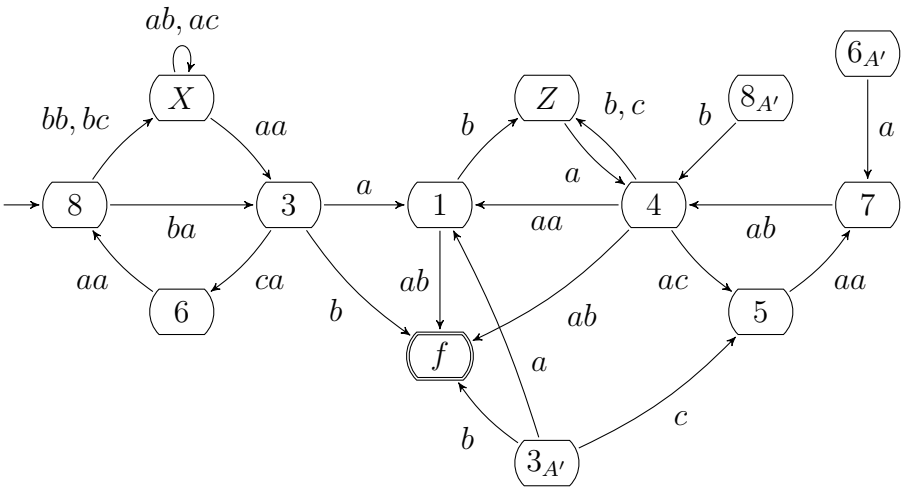


FIGURE 4.19 – Une complétion possible de l'orbite maximale de A'

4.5.1.2 La procédure d'élimination

Nous montrons maintenant comment calculer un automate à blocs $\mathcal{L}$-équivalent à partir d'une sélection d'orbites maximales.

Soient M l'AFD minimal d'un ABD A, et $(K_1, K_2, \ldots, K_l)$ l'ensemble partiellement ordonné équivalent à $\{\Omega(O) \mid O \text{ est une orbite de } A\}$, tel que K_{i+j} ne soit pas accessible depuis K_i. Remarquons que K_l contient l'état initial de M (car M est émondé). La procédure d'élimination se compose des étapes suivantes :

1. Sélectionner $(O_1, O_2, \ldots, O_l)$ maximales dans A tel que $\Omega(O_j) = K_j$.
2. Calculer un automate à blocs $\mathrm{Comp}(A)$ en complétant chaque orbite O_i de A avec les états de $Q_i = \Theta(O_i) \cap \Phi(Q_A)$. Alors $C_i = O_i \cup Q_i$ est une *orbite complétée*.
3. Calculer un automate à blocs $\mathrm{Link}(A)$ en reliant ensemble chaque orbite complétée de $\mathrm{Comp}(A)$ tout en préservant la transversalité.
4. Calculer un automate à blocs $\mathrm{Slim}(A)$ à partir de $\mathrm{Link}(A)$ en changeant l'état initial par un de ceux de C_l équivalent à i_A, et en ne conservant que les états des orbites complétées.

Exemple 4.35. L'automate A' en Figure 4.17 n'a que des orbites compactes. L'automate M en Figure 4.18 est son AFD minimal, et a trois orbites dont $K_1 = \{f\}$, $K_2 = \{1, 3, 4, 5, 6, 7, 8, X\}$ et $K_3 = \{2\}$. Cette dernière ne nous intéresse pas puisqu'elle n'a pas d'équivalent dans A'. Tout d'abord, nous sélectionnons les orbites maximales suivantes de A' : $O_1 = \{f\}$ et $O_2 =$

$\{1, 4, 5, 7, Z\}$. Nous obtenons que $\Theta(O_1) = \emptyset$ et $\Theta(O_2) \cap \Phi(Q_{A'}) = \{3, 6, 8\}$. Ainsi, une des complétions des orbites de A' est présentée en Figure 4.19, et un automate compact $\mathrm{Slim}(A')$ est présenté en Figure 4.20. Remarquons qu'afin d'obtenir un automate $\mathcal{L}$-équivalent à l'automate de départ, cette méthode crée un automate non nécessairement émondé.

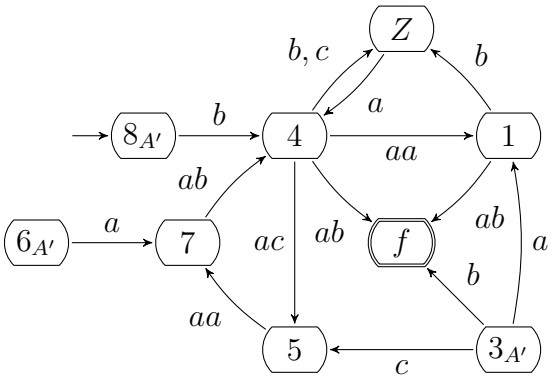


FIGURE 4.20 – L'automate $\mathrm{Slim}(A')$

Nous étudions à présent les propriétés préservées par l'élimination d'orbites d'un automate à blocs :

Proposition 4.36. *Soient A un ABD k-blocs validant le BW-test, et B un automate obtenu par la procédure d'élimination d'orbites. Alors :*

1. *B est déterministe,*
2. *B est k-blocs,*
3. *B valide le BW-test,*
4. *B est $\mathcal{L}$-équivalent à A,*
5. *$\mathcal{F}_B$ est un sous-ensemble de $\mathcal{F}_A$,*
6. *si toutes les orbites de A sont compactes, alors B est compact.*

Démonstration. Soient $\mathrm{Comp}(A)$ et $\mathrm{Link}(A)$ les automates décris plus haut pour calculer B.

D'après la Proposition 4.33, $\mathrm{Comp}(A)$ est déterministe, k-blocs, $\mathcal{L}$-équivalent à A et valide le BW-test. De plus, $\mathcal{F}_{\mathrm{Comp}(A)} = \mathcal{F}_A$.

Pour relier les orbites complétées entre elles, les transitions de sortie doivent être redirigées vers des états équivalents. Par construction, pour chaque orbite complétée C_i de $\mathrm{Comp}(A)$ et pour toute transition de sortie (o, b, q) de C_i, il existe exactement une orbite $C_{j<i}$ contenant au moins

un état équivalent à q. La structure de chaque orbite de Comp(A) et leur transversalité sont donc préservées dans Link(A). Comme Comp(A) valide le BW-test, alors Link(A) le valide donc également. Par ailleurs, le fait de rediriger les transitions vers des états équivalents préservent les langages droits, le déterminisme, la taille maximale des blocs et l'ensemble des étiquettes finales. Enfin, en reliant ces orbites complétées les unes après les autres, nous obtenons à chaque étape un ABD $\mathcal{L}$-équivalent au précédent.

En changeant l'état initial par un de ceux de l'orbite complétée C_l, les états n'appartenant pas à une des orbites complétées sont alors inaccessibles. En retirant ces états, nous préservons le déterminisme (1), la taille maximale des blocs (2), et obtenons que $\mathcal{F}_B$ est inclus dans $\mathcal{F}_{\text{Link}(A)} = \mathcal{F}_A$(5). Par ailleurs, le fait de retirer entièrement des orbites préserve également le BW-test (3). Enfin, comme les langages droits des états des orbites complétées sont préservés et que pour tout état de Link(A), il existe un état équivalent dans une orbite complétée, alors B est $\mathcal{L}$-équivalent à Link(A)(4).

Pour terminer, nous supposons que toutes les orbites de A sont compactes. Pour toutes deux orbites sélectionnées distinctes O_i et O_j de A, $\Omega(O_i) \neq \Omega(O_j)$. Donc tout état de l'une de ces orbites n'est pas équivalent à un état de l'autre. Comme la structure des orbites sélectionnées est préservée ainsi que les langages droits de leurs états, toutes les orbites de B sont également compactes. De plus, comme chaque orbite sélectionnée est complétée par des états qui n'ont pas d'équivalent dans cette orbite, alors deux états distincts de B ne sont pas équivalents (6). ■

4.5.2 La procédure de compaction

Il nous faut maintenant pouvoir compacter récursivement les différentes orbites non compactes d'un ABD. L'automate obtenu est appelé un *compacté* de l'automate de départ. Soit A un ABD k-blocs validant le BW-test. Nous procédons de la manière suivante :

— Si toutes les orbites de A sont compactes, nous appliquons la procédure d'élimination d'orbites à A.

— Sinon, il existe une orbite non-compacte O de A et un couple (b, s) dans l'ensemble de synchronisation de O. Nous procédons alors récursivement à la compaction de l'automate $A_s^{-(b,s)}$. Les transitions de synchronisation sont alors replacées dans l'automate compact obtenu (vers son état initial), pour ensuite le substituer à l'orbite O dans A. L'automate obtenu est alors récursivement compacté.

Nous commençons par étudier les sous-opérations nécessaires puis nous terminons par la preuve de la procédure.

4.5.2.1 Extraire et substituer une orbite

Pour compacter une orbite, nous commençons par calculer un de ses automates orbitaux. Cette extraction préserve le déterminisme, la taille maximale des blocs et la validation du BW-test. Par ailleurs, la preuve de la Proposition 3.53 (en page 73) peut être étendue naturellement aux automates à blocs non-ambigus, dont font partie les ABD d'après le Lemme 4.3 :

Corollaire 4.37. *Soit O une orbite transverse d'un ABD A. Alors pour tout deux états p et q de O, leurs langages droits sont égaux si et seulement si leurs langages orbitaux le sont.*

Nous étudions maintenant l'opération de substitution d'une orbite. Soient A un automate à blocs satisfaisant la propriété orbitale, q un état de A et B un automate à blocs $\mathcal{L}$-équivalent à A_q. Puisque B et A_q sont $\mathcal{L}$-équivalents, il existe au moins une fonction h associant chaque état de A_q à son équivalent dans B. Une *substitution de* $\mathrm{O}_A(q)$ *par* B construit un automate à blocs C défini comme suit :

- $\Sigma_C = \Sigma_A \cup \Sigma_B$
- $Q_C = (Q_A \setminus \mathrm{O}(q)) \cup Q_B$
- $I_C = \{h(i_A)\}$ si $i_A \in \mathrm{O}(q)$, $\{i_A\}$ sinon
- $F_C = (F_A \setminus \mathrm{O}(q)) \cup F_B$ si $F_A \cap \mathrm{O}(q) \neq \emptyset$, F_A sinon
- $\delta_C = \begin{array}{l} (\delta_A \setminus (Q_A \times \Gamma_A \times \mathrm{O}(q)) \setminus (\mathrm{O}(q) \times \Gamma_A \times Q_A)) \\ \quad \cup \delta_B \\ \quad \cup \{(p, b, h(o)) \mid (p, b, o) \in \delta^{\mathrm{in}}(\mathrm{O}(q))\} \\ \quad \cup \{(f_B, b, p) \mid f_B \in F_B \wedge \exists (o, b, p) \in \delta^{\mathrm{out}}(\mathrm{O}(q))\} \end{array}$

Exemple 4.38. Soient A l'automate représenté en Figure 4.14 (page 102), B l'automate orbital de l'état Y de A représenté en Figure 4.21, et B' un automate $\mathcal{L}$-équivalent à B représenté en Figure 4.22. La substitution de l'orbite de l'état Y par B' dans A produit l'automate A' en Figure 4.17 (page 106).

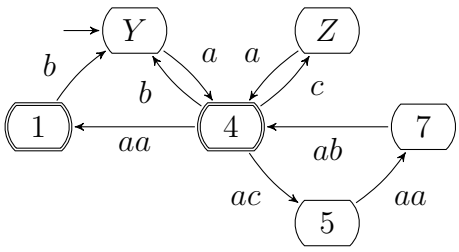


FIGURE 4.21 – L'automate B

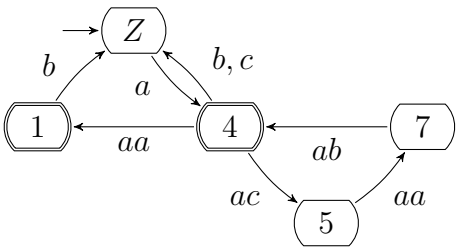


FIGURE 4.22 – L'automate B'

Nous étudions les propriétés préservées par l'opération de substitution :

Proposition 4.39. *Soit q un état d'un automate k-blocs déterministe A validant le BW-test. Soit B un automate k-blocs déterministe $\mathcal{L}$-équivalent à A_q, tel que $\mathcal{F}_B$ soit inclus dans $\mathcal{F}_{A_q}$. Soit C un automate à blocs obtenu par la substitution de l'orbite de q par B dans A, alors :*

1. *C est k-blocs,*
2. *$\mathcal{F}_C$ est est inclus dans $\mathcal{F}_A$,*
3. *C est déterministe,*
4. *C est $\mathcal{L}$-équivalent à A,*
5. *si B valide le BW-test, alors C aussi,*
6. *si B est compact et satisfait la propriété orbitale, alors toutes les orbites de B sont compactes dans C.*

Démonstration. (1) : Puisque A et B sont k-blocs, C l'est aussi.

(2) : Deux cas peuvent se présenter. Soit F_B et F_C sont disjoints, et donc $\mathcal{F}_C = \mathcal{F}_A$. Soit F_B est inclus dans F_C, et donc les portes de sortie de $\mathrm{O}_A(q)$ sont finales et les étiquettes de leurs transitions de sortie sont incluses dans $\mathcal{F}_A$. Puisque $\mathcal{F}_B$ est inclus dans $\mathcal{F}_{A_q}$, alors $\mathcal{F}_C$ est inclus dans $\mathcal{F}_A$.

(3) : Comme A et B sont déterministes, il suffit de vérifier les transitions de sortie des états de F_B dans C. Puisque $\mathcal{F}_B$ est inclus dans $\mathcal{F}_{A_q}$ et que les transitions sortant des états de B et entrant dans ceux de A ont les mêmes étiquettes que les transitions de sortie de $\mathrm{O}_A(q)$, alors C est déterministe.

(4) : Les langages droits des états de B dans C correspondent à la concaténation de leurs langages droits dans B avec le langage externe de $\mathrm{O}_A(q)$. Puisque B et A_q sont $\mathcal{L}$-équivalents, la famille de langages droits de B est égale à l'ensemble des langages orbitaux de $\mathrm{O}_A(q)$, et l'ensemble des langages droits des états de B dans C est le même que celui des états de $\mathrm{O}_A(q)$. En redirigeant les transitions d'entrée de $\mathrm{O}_A(q)$ vers des états équivalents dans B, nous obtenons que A et C sont bien $\mathcal{L}$-équivalents.

(5) : Supposons que B valide le BW-test. La structure des orbites de B étant préservée dans C, tout comme celle des orbites de A distinctes de $\mathrm{O}_A(q)$, leurs automates orbitaux valident donc le BW-test. En ajoutant des transitions sortantes à tous les anciens états finaux de B, la transversalité des orbites de C est préservée. De plus, le fait de rediriger les transitions d'entrée de $\mathrm{O}_A(q)$ ayant la même cible vers le même état de B préserve également la transversalité des orbites de A présentes dans C. Donc C valide bien le BW-test.

(6) : Supposons que B soit compact et satisfasse la propriété orbitale. Ses orbites sont donc également transverses dans C. Puisque C est déterministe, d'après le Corollaire 4.37, les orbites de B sont bien compactes dans C. ■

L'opération de substitution permet donc de réduire d'une unité le nombre d'orbites non compactes. Nous aurons alors besoin de l'appliquer séquentiellement sur toutes les orbites non compactes.

4.5.2.2 Retirer et ajouter les transitions de synchronisation

Afin de compacter les automates orbitaux calculés précédemment tout en suivant le BW-test, nous retirons l'un de ses couples de synchronisation, pour le recréer après compaction. Remarquons que ce retrait, comme l'extraction d'orbite, préserve le déterminisme, n'augmente pas la taille maximale des blocs et préserve le BW-test. Par ailleurs, la preuve de la Proposition 3.55 (en page 75) peut également être étendue aux ABD :

Corollaire 4.40. *Soient A un ABD et (b, s) un élément de $\mathcal{S}_A$. Alors, pour tout deux états p et q de A, leurs langages droits sont égaux si et seulement si leurs langages de coupure par (b, s) le sont.*

Nous étudions alors l'opération inverse à la coupure : l'ajout d'un couple de synchronisation. Soit A un automate à blocs. Un bloc w est *A-vacant* s'il existe un état q_w de A tel qu'aucun état final de A n'a de transition sortante étiquetée par w vers q_w. L'*ensemble de vacance* de A est noté $\mathcal{V}_A = \{(w, q_w) \mid \forall f \in F, (f, w, q_w) \notin \delta\}$. Soit (w, q_w) un élément de $\mathcal{V}_A$. L'ajout de v dans A, noté $A^{+(w,q_w)}$, est alors construit par l'ajout pour chaque état final f de A, de la transition (f, w, q_w). Cette opération est ensuite naturellement étendue à un sous-ensemble de $\mathcal{V}_A$. Remarquons que si (a, p) appartient à $\mathcal{S}_A$, alors $(A^{-(a,p)})^{+(a,p)} = A$, et inversement, si (b, q) appartient à $\mathcal{V}_A$, alors $(A^{+(b,q)})^{-(b,q)} = A$.

Exemple 4.41. Soient B l'automate représenté en Figure 4.21, C la coupure par (b, Y) de B représenté en Figure 4.23, et C' un automate $\mathcal{L}$-équivalent à C représenté en Figure 4.24. Alors l'automate B', représenté en Figure 4.22 (page 110), est obtenu à partir de C' en ajoutant les transitions de synchronisation étiquetées par b vers l'état Z.

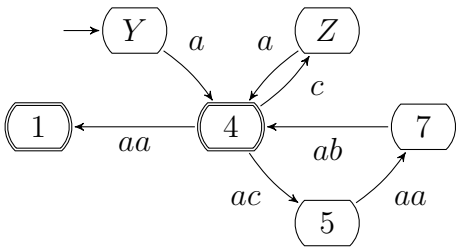


FIGURE 4.23 – L'automate C

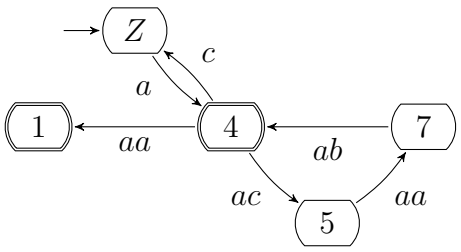


FIGURE 4.24 – L'automate C'

Proposition 4.42. *Soit A un automate k-blocs déterministe tel que (b, s) appartienne à $\mathcal{S}_A$. Soit B un automate k-blocs déterministe $\mathcal{L}$-équivalent à $A^{-(b,s)}$ tel que $\mathcal{F}_B$ soit un sous-ensemble de $\mathcal{F}_{A^{-(b,s)}}$. Soit s' un état de B équivalent à s dans $A^{-(b,s)}$. Alors :*

1. *b est vacant dans B,*
2. *$B^{+(b,s')}$ est k-blocs,*
3. *$\mathcal{F}_{B^{+(b,s)}}$ est un sous-ensemble de $\mathcal{F}_A$,*
4. *$B^{+(b,s')}$ est déterministe,*
5. *$B^{+(b,s')}$ est $\mathcal{L}$-équivalent à A,*
6. *si B valide le BW-test, alors $B^{+(b,s')}$ aussi,*
7. *si B est compact, alors $B^{+(b,s')}$ aussi.*

Démonstration. (1) : Comme A est déterministe, $A^{-(b,s)}$ l'est aussi. Donc aucun mot du langage droit de ses états finaux ne commence par le bloc b. Il en est donc de même pour les états finaux de B, dont b est alors vacant.

(2) : Comme A est k-blocs, b est de longueur au plus k. Nous concluons par le fait que B est également k-blocs.

(3) : Nous avons $\mathcal{F}_{B^{+(b,s)}} = \mathcal{F}_B \cup \{b\}$ et $\mathcal{F}_A = \mathcal{F}_{A^{-(b,s)}} \cup \{b\}$. Comme $\mathcal{F}_B$ est un sous-ensemble de $\mathcal{F}_{A^{-(b,s)}}$, $\mathcal{F}_{B^{+(b,s)}}$ est un sous-ensemble de $\mathcal{F}_A$.

(4) : Puisque A est déterministe et que (b, s) est A-synchronisant, pour tout mot w de $\mathcal{F}_A \setminus \{b\}$, b n'est pas préfixe de w et w n'est pas préfixe de b. Cela reste vrai pour tout mot w de $\mathcal{F}_B$ qui est un sous-ensemble de $\mathcal{F}_{A^{-(b,s)}}$. Comme B est déterministe, alors $B^{+(b,s')}$ l'est aussi.

(5) : Soit p un état de B. D'après le Lemme 3.41, nous avons $\mathrm{L}_{B^{+(b,s')}}(p) = \mathrm{L}_B(p) \cdot (\{b\} \cdot \mathrm{L}_B(s'))^*$. Puisque B est $\mathcal{L}$-équivalent à $A^{-(b,s)}$, il existe un état p' de $A^{-(b,s)}$ qui est équivalent à p dans B. Cet état p' est donc un état de A et $\mathrm{L}_A(p') = \mathrm{L}_{A^{-(b,s)}}(p') \cdot (\{b\} \cdot \mathrm{L}_{A^{-(b,s)}}(s))^*$. Puisque s' dans B est équivalent à s dans $A^{-(b,s)}$, nous obtenons que $\mathrm{L}_A(p') = \mathrm{L}_B(p) \cdot (\{b\} \cdot \mathrm{L}_B(s'))^*$, et donc que $\mathrm{L}_A(p') = \mathrm{L}_{B^{+(b,s')}}(p)$. De plus, pour tout état q de $A^{-(b,s)}$, il existe un état q' dans B qui est équivalent à q dans $A^{-(b,s)}$. Avec le même raisonnement, nous concluons que $\mathrm{L}_{B^{+(b,s')}}(q') = \mathrm{L}_A(q)$. Ainsi, $B^{+(b,s')}$ est $\mathcal{L}$-équivalent à A.

(6, 7)) : Puisque A est déterministe, b n'appartient pas à $\mathcal{F}_{A^{-(b,s)}}$. Ce qui signifie que b n'appartient pas à $\mathcal{F}_B$, que (b, q) n'appartient pas à $\mathcal{V}_B$ et donc que $B = (B^{+(b,s')})^{-(b,s')}$. Par Définition du BW-test, si B valide le BW-test, alors $B^{+(b,s')}$ aussi. Enfin, d'après (3) et le Corollaire 4.40, si B est compact, alors $B^{+(b,s')}$ l'est aussi. ■

4.5.2.3 Preuve de la décidabilité

Nous pouvons alors faire la preuve des propriétés de notre procédure de compaction :

Proposition 4.43. *Soient A un ABD k-blocs validant le BW-test, et D le compacté de A. Alors*

1. *D est déterministe,*
2. *D est k-blocs,*
3. *D valide le BW-test,*
4. *D est $\mathcal{L}$-équivalent à A,*
5. *$\mathcal{F}_D$ est un sous-ensemble de $\mathcal{F}_A$,*
6. *D est compact.*

Démonstration. Si toutes les orbites de A sont compactes, alors D est l'automate obtenu par la procédure d'élimination des orbites. La Proposition 4.36 permet alors de conclure.

Sinon, il existe une orbite O de A qui n'est pas compacte. Cette orbite n'est donc pas triviale et puisque A valide le BW-test, l'ensemble de synchronisation de O n'est pas vide. Soient (b, s) un élément de l'ensemble de synchronisation de O, et B l'automate orbital de s. Puisqu'il s'agit d'un automate orbital obtenu à partir de A, il est donc déterministe, k-blocs, et valide le BW-test. Soit C la coupure par (b, s) de B. Puisqu'il est obtenu à partir de B, alors il est déterministe, k-blocs, et valide le BW-test. Soit C' un compacté de C. Puisque ce dernier est structurellement plus petit que A (d'au moins une transition), il vérifie les points 1 à 6 par rapport à C. Soit B' l'automate obtenu après l'ajout de transitions de synchronisation étiquetées par b vers $i_{C'}$ dans C'. D'après la Proposition 4.42, B' vérifie les points 1 à 6 par rapport à B. Soit A' l'automate obtenu par la substitution de O par B' dans A. D'après la Proposition 4.39, A' vérifie les points 1 à 6 par rapport à A. Par ailleurs, A' a une orbite non triviale de moins que A. L'automate D est alors un compacté de A' qui, par induction sur le nombre d'orbites non-compactes de A' par rapport à A, vérifie les points 1 à 6. ■

Remarquons que puisque le BW-test se termine toujours, soit par l'échec, soit en atteignant des automates acycliques, la procédure de compaction se termine également toujours. En effet, un automate acyclique ne contient que des orbites triviales qui sont donc compactes, impliquant la fin de la suite d'appels récursifs.

Exemple 4.44. Nous résumons l'exécution de la procédure de compaction sur l'automate A (Figure 4.14 en page 102). Il est 2-blocs, déterministe et valide le BW-test, mais n'est pas compact et $\mathrm{O}_A(Y)$ non plus. Nous calculons alors $B = A_Y$ (Figure 4.21 en page 110) dont l'ensemble de synchronisation est $\{(b, Y)\}$. La coupure par $\{(b, Y)\}$ de B nous donne l'automate $C = B^{-(b,Y)}$ (Figure 4.23 en page 112). Ce dernier n'est pas compact mais toutes ses orbites le sont. Nous en déduisons un compacté C' (Figure 4.24 en page 112). L'automate B' (Figure 4.22 en page 110) est obtenu en replaçant les transitions de synchronisation par b vers l'état initial Z. En substituant $\mathrm{O}_A(Y)$ par B' dans A, nous obtenons l'automate A' (Figure 4.17 en page 106). Puisque A' n'a que des orbites compactes, un compacté peut être calculé (Figure 4.20 en page 108).

Les automates obtenus par la procédure de compaction ne sont pas nécessairement émondés. Toutefois, l'émondage préserve le déterminisme, la taille maximale des blocs et la validité du BW-test (car les états d'une orbite sont soit tous accessibles, soit tous non-accessibles). En appliquant la procédure de compaction , nous pouvons alors préciser le Théorème 4.27 :

Théorème 4.45. *Un langage rationnel est k-blocs déterministe si et seulement s'il est reconnaissable par un automate k-blocs, déterministe, compact, émondé et validant le BW-test.*

Ce qui, d'après le Théorème 4.19, permet de conclure que :

Théorème 4.46. *Le k-blocs déterminisme d'un langage rationnel est décidable à partir de son AFD minimal.*

4.6 Hiérarchie et inclusion stricte

Grâce à la décidabilité du k-blocs déterminisme, nous pouvons refaire la preuve du Théorème 4.28 en définissant notre propre famille de langages.

Définition 4.47. Soit n un entier naturel. L'automate $B_n = (\{a, b, c\}, Q, I, F, \delta)$ est défini de la manière suivante :

- $Q = \{f\} \cup \{\alpha_j, \beta_j \mid 1 \leq j \leq n\}$,
- $I = \{\beta_n\}$,
- $F = \{f\} \cup \{\alpha_n, \beta_n\}$,
- $\delta = \{(\beta_n, a, \alpha_n), (\beta_1, b, f), (\alpha_n, a, \alpha_n), (\alpha_1, b, f), (\alpha_1, c, \beta_n)\}$
  $\cup \{(\alpha_j, b, \alpha_{j-1}), (\beta_j, b, \beta_{j-1}) \mid 2 \leq j \leq n\}$.

La famille d'automates $\mathcal{B}$ est alors définie par $\{B_j \mid j \in \mathbb{N} \setminus \{0, 1\}\}$.

Un représentant de cette famille est illustrée en Figure 4.25.

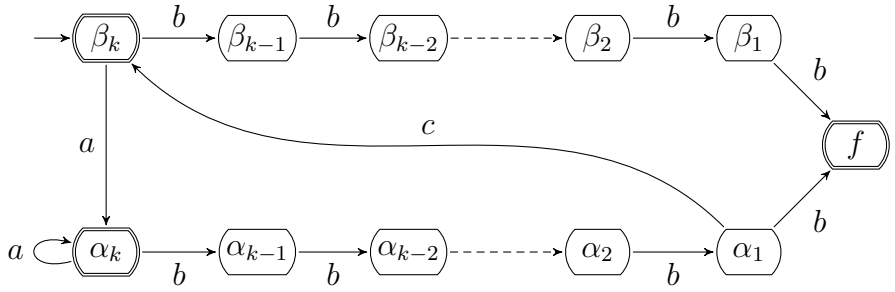


FIGURE 4.25 – Un automate B_k de la famille $\mathcal{B}$

Par construction, tous les membres de cette famille sont déterministes et émondés. Un des automates pouvant être calculés à partir de l'automate des k-transitions de $\mathrm{L}(B_k)$ est représenté en Figure 4.26.

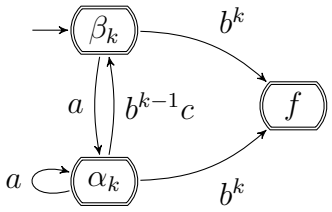


FIGURE 4.26 – Un automate validant le BW-test obtenu à partir de B_k

Ce dernier valide le BW-test et permet d'en déduire l'expression k-blocs déterministe $(a(\varepsilon+[b^{k-1}c]))^*(\varepsilon+[b^k])$. Il faut ensuite montrer que ce langage n'est pas $(k-1)$-blocs déterministe.

Nous commençons par vérifier qu'on ne peut pas obtenir un ABD validant le BW-test à partir de l'automate des $(k-1)$-transitions. L'unique orbite non triviale de B_k contient tous les états α et l'état β_k, et ses portes de sortie sont les états α_1, α_k et β_k. Cette orbite n'est donc pas transverse. Puisque α_k et β_k sont finaux, il est nécessaire que ces deux états deviennent les seules portes de sortie de leur orbite. Or, pour cela, il faut que l'état α_1 devienne inaccessible depuis α_k. Ce faisant, l'état α_k aura nécessairement une transition vers f étiquetée par un bloc de longueur k.

Il reste alors à vérifier que tous les automates de la famille $\mathcal{B}$ reconnaissent des langages qui leur sont propres. Pour cela, il suffit de montrer qu'il s'agit d'AFD minimaux. L'état final f étant le seul état sans successeur, il n'est équivalent à aucun autre. Ses seuls prédécesseurs directs sont

α_1 dont le langage droit est fini et β_1 dont le langage droit est infini. Ils ne sont donc équivalents à aucun autre. En appliquant la même logique pour tout couple d'états (α_i, β_i), nous obtenons que les langages droits sont tous distincts. Ces automates sont donc des AFD minimaux ayant tous un nombre d'états qui leur sont propres, et reconnaissant donc des langages qui leur sont propres.

Proposition 4.48. *Pour tout entier k strictement supérieur à 1, le langage $\mathrm{L}(B_k)$ est k-blocs déterministe sans être $(k-1)$-blocs déterministe.*

Nous terminons cette partie en démontrant que la famille des langages blocs déterministes est strictement incluse dans celle des rationnels. Pour cela, nous utilisons une propriété simple des mots sur un alphabet unaire : pour tout couple de mots, l'un est nécessairement un préfixe de l'autre. Ce qui nous donne sur les expressions rationnelles :

Lemme 4.49. *Soit E une expressions rationnelle à blocs sur un alphabet unaire. Si E est déterministe, alors $|\mathrm{First}(E^\sharp)| \leq 1$ et pour toute position x dans Π_E, $|\mathrm{Follow}(E^\sharp, x)| \leq 1$.*

Remarquons que si une expression rationnelle classique (à lettres) présente les deux propriétés ci-dessus, alors elle est déterministe. Or, remplacer chaque bloc $[b_1 \cdots b_n]$ par l'expression $b_1 \cdots b_n$ préserve et reflète ces propriétés. Ce qui nous permet d'en conclure que :

Théorème 4.50. *Si un langage rationnel unaire est blocs déterministe, alors il est déterministe.*

Nous utilisons alors la réciproque de ce Théorème sur l'AFD minimal du langage $\neg((aaa)^*)$, représenté en Figure 3.12, pour conclure :

Théorème 4.51. *Il existe des langages rationnels qui ne sont pas blocs déterministes.*

4.7 Conclusion

Nous avons pu redémontrer l'existence d'une hiérarchie stricte au sein de la famille des langages blocs déterministes, et démontrer son inclusion stricte dans la famille des rationnels. Nous n'avons toutefois pu redémontrer que la décidabilité du k-blocs déterminisme. L'existence d'une borne supérieure k tel qu'un langage rationnel soit k-blocs déterministe ou ne soit pas blocs déterministe reste donc ouverte.

Chapitre 5

Langages localement prédictibles

Brüggemann-Klein et Wood [14] ont eux-mêmes évoqué une généralisation des expressions rationnelles déterministes, consistant à utiliser une fenêtre de longueur constante k sur le mot en entrée. La prochaine position dans l'expression est alors déterminée par rapport aux facteurs de longueur k dénotés à partir de la position courante, de sorte qu'il ne doit exister au plus qu'un seul successeur direct par lequel passer pour lire entièrement le contenu de la fenêtre. Par ailleurs, à la différence des expressions rationnelles à blocs où ces derniers sont lus entièrement, la fenêtre n'est ici décalée que d'un symbole à la fois, de sorte que les symboles suivants servent à nouveau pour déterminer la prochaine position.

Han et Wood [36] ont formalisé cette propriété sur les expressions rationnelles et les automates, puis ont étudié les relations existant avec la famille des langages blocs déterministes. Toutefois, indépendamment de ces travaux et antérieurement, une famille d'automates plus générale avait été présentée. Nous abordons donc ce Chapitre par le contexte dans lequel se situe la famille des automates localement prédictibles avant de présenter nos résultats sur la famille des langages localement prédictibles.

5.1 Historique

5.1.1 Automates prédictibles

Gerede *et al.* [29] et Dang *et al.* [26] ont étudié le paradigme de programmation des services en réseau, permettant de réutiliser des services déjà existants pour les composer de manière automatisée et en créer de nouveaux. Ces services, représentés par des automates, doivent pouvoir être sélectionnés spécifiquement parmi une liste pour exécuter une séquence d'actions

décrivant un mot du langage reconnu par cet automate. Le choix de sélectionner tel ou tel automate se fait à l'aide d'un délégateur considérant les k prochaines actions. La question était alors de savoir si un tel délégateur existe pour un système donné. Ravikumar et Santean [51] ont formalisé cette notion de délégateur sur les automates. Ces derniers peuvent être non déterministes mais ont une structure tel qu'en utilisant un préfixe du mot en entrée, d'une taille maximale fixée, il est possible de prédire quelle transition emprunter et ainsi retrouver une forme de déterminisme. Les auteurs ont également donné les premiers résultats sur la décidabilité de cette propriété dans le cas des automates non-ambigus. Résultats qui ont ensuite été généralisés par Löding et Repke [43] à tout automate fini.

Un k-délégateur est ici une fonction permettant, à partir des k prochaines lettres d'un mot en entrée, de choisir une transition de façon déterministe, et de préserver le langage reconnu. Mais à la différence du bloc déterminisme, seule la première lettre du mot en entrée est lue et non le préfixe entier, les lettres suivantes étant réutilisées pour décider des prochaines transitions. Nous définissons un délégateur d'un automate A comme un sous-ensemble de $Q_A \times \Sigma_A^* \times Q_A$, équivalent à une fonction de $Q_A \times \Sigma_A^*$ vers $\mathcal{P}(Q_A)$, qui est elle-même étendue de $\mathcal{P}(Q_A) \times \Sigma_A^*$ vers $\mathcal{P}(Q_A)$. Formellement, cela se définit comme suit :

Définition 5.1 ([43])**.** Soient A un automate avec un unique état initial i_A, Δ un sous-ensemble de $Q_A \times \Sigma_A^* \times Q_A$ et k un entier strictement positif. Alors Δ est un *k-délégateur* de A si les trois conditions suivantes sont vérifiées :

1. pour tout (p, w, q) dans Δ, alors $1 \leq |w| \leq k$ et $(p, w[1], q)$ est une transition dans δ_A,
2. pour tout deux éléments distincts (p, w_1, q_1) et (p, w_2, q_2) dans Δ, alors w_1 est différent de w_2,
3. un mot w est dans $\mathrm{L}(A)$ si et seulement si l'intersection de $\Delta^r(i_A, w)$ avec F_A n'est pas vide.

La fonction Δ^r est la clôture dite recouvrante de Δ tel que pour tout état q de A, $\Delta^r(q, \varepsilon) = \{q\}$, et pour toute lettre a et tout mot w de longueur quelconque, $\Delta^r(q, aw) = \Delta^r(\Delta(q, aw[1, n]), w)$ avec $n = \mathrm{Min}(|w|, k-1)$.

Comme les délégateurs ne traitent que des transitions et que l'on souhaite obtenir un comportement déterministe, les automates sur lesquels ceux-ci sont définis n'ont qu'un seul état initial. Un automate est alors dit *k-prédictible* s'il existe un k-délégateur sur ce dernier, et *prédictible* s'il est k-prédictible pour un certain k.

Exemple 5.2. Soit A l'automate représenté en Figure 5.1, reconnaissant le langage dénoté par $a + (a + b)^*(b + aa)$. Le non déterminisme de A intervient uniquement au moment de choisir une des deux transitions sortant de i par a. Il est toutefois possible de définir un 2-délégateur Δ contenant au moins les éléments (i, a, f), (i, aa, i) et (i, ab, i) ; les autres éléments se déduisant immédiatement. Remarquons qu'il est possible de définir un autre délégateur en remplaçant dans le premier l'élément (i, ab, i) par (i, ab, f).

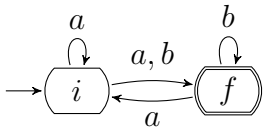


FIGURE 5.1 – Un automate 2-prédictible

La lecture d'un mot peut alors se faire à l'aide d'une fenêtre glissante. Cette dernière doit au préalable être remplie entièrement ou, à défaut, contenir le mot entier. Puis elle sera décalée un nombre de fois équivalent à la longueur du mot. En utilisant des arguments similaires à ceux utilisés pour le Théorème 4.4 (en page 81), nous pouvons en déduire que :

Théorème 5.3. *Soient A un automate k-prédictible et w un mot. L'appartenance de w au langage reconnu par A est décidable en temps $O(|w| + k)$.*

Nous nous intéressons maintenant à la décidabilité de cette propriété. Löding et Repke déduisent de la Définition 5.1 une condition nécessaire et suffisante pour l'existence d'un k-délégateur pour un automate. Soient p un état de A accessible par le délégateur, a une lettre étiquetant une des transitions sortant de p et w un mot de longueur $k - 1$. Afin de préserver le langage reconnu, il faut qu'il existe une transition (p, a, q) tel que pour tout mot awv du langage droit de p, le mot wv soit dans le langage droit de q. Ou, autrement dit :

Lemme 5.4 ([43])**.** *Soit A un automate avec un unique état initial i_A. Alors il existe un k-délégateur de A si et seulement s'il existe un sous-ensemble Q' de Q tel que :*

1. *i_A appartient à Q',*
2. *pour tout état p de Q', toute lettre a et tout mot w de longueur $k - 1$, il existe un état q de Q' tel que la transition (p, a, q) est dans δ_A et $(aw)^{-1}\mathrm{L}(p) = w^{-1}\mathrm{L}(q)$.*

Exemple 5.5. Soit A l'automate représenté en Figure 5.2, reconnaissant le langage dénoté par ab^*. Le non déterminisme de A intervient uniquement au moment de choisir une des deux transitions sortant de i par a. Or, le langage droit de 1 ne contient que les mots de longueur paire et celui de 3 que les mots de longueur impaire. Selon le Lemme précédent, il n'existe donc pas de délégateur pour A.

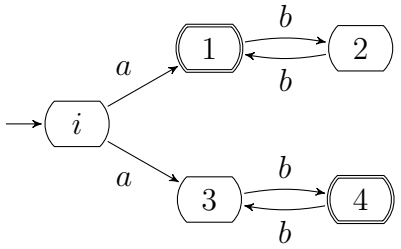


FIGURE 5.2 – Un automate non prédictible

Les auteurs utilisent ce Lemme pour prouver la complexité du problème de l'existence d'un k-délégateur pour un automate et un k fixé :

Théorème 5.6 ([43]). *Soit A un automate avec un unique état initial. L'existence d'un k-délégateur de A est décidable en temps $O(k \times |\Sigma_A|^k \times |Q_A|^2)$.*

Enfin, ils démontrent l'existence d'une borne maximale pour la recherche de l'indice d'un délégateur et ainsi, que le problème de l'existence d'un délégateur pour un automate est décidable :

Théorème 5.7 ([43]). *Si un automate A est prédictible, alors il est $(2^{n^2}+1)$-prédictible, avec $n = |Q_A|$.*

Nous notons toutefois que les auteurs ne montrent pas que cette borne puisse être atteinte.

5.1.2 Automates localement prédictibles

Indépendamment des travaux sur les automates prédictibles, Han et Wood [36] ont étudié la généralisation proposée par Brüggemann-Klein et Wood. Ici, il faut pouvoir décider si un k-délégateur peut être construit uniquement à partir des chemins de longueur au plus k partant de chaque état. Cette propriété ne se base donc pas sur les mots des langages droits.

Ce faisant, pour chaque état q et tout mot aw de longueur k, l'état d'arrivée de la transition retournée devra alors être le seul successeur direct de q permettant de lire w, contrairement aux automates prédictibles qui en autorisent plusieurs.

Soient A un automate, q un état de A et k un entier. L'ensemble $E_k(q)$ des *étiquettes de* q contient les étiquettes des chemins de longueur k partant de q. Nous pouvons alors définir cette propriété :

Définition 5.8. Soit k un entier strictement positif. Un automate est *k-localement prédictible* s'il a au plus un état initial et si pour toutes transitions distinctes (p, a, q_1) et (p, a, q_2), l'intersection de $E_{k-1}(q_1)$ avec $E_{k-1}(q_2)$ est vide. Un automate est dit localement prédictible s'il est k-localement prédictible pour un certain k.

Nous remarquons que cette Définition étend bien celle du déterminisme d'un automate puisqu'un automate 1-localement prédictif est déterministe. En effet, étant donné que pour tout état, l'ensemble de ses étiquettes de longueur 0 est toujours égal à $\{\varepsilon\}$, s'il existe deux transitions distinctes (p, a, q_1) et (p, a, q_2) dans l'automate, alors l'intersection de $E_0(q_1)$ avec $E_0(q_2)$ est non vide et l'automate n'est pas 1-localement prédictif. Par ailleurs, un automate k-localement prédictible est k-prédictible.

Exemple 5.9. Si nous considérons l'automate 2-prédictible représenté en Figure 5.1, nous remarquons que ce dernier n'est pas localement prédictible. En effet, pour tout entier k, l'intersection de $E_k(i)$ avec $E_k(f)$ n'est pas vide.

Considérons maintenant l'automate représenté en Figure 5.3. Comme ce dernier n'est pas déterministe, il n'est pas 1-localement prédictible. De plus, le non-déterminisme intervient uniquement au moment de choisir une des deux transitions sortant de i par a. Comme l'intersection de $E_1(1)$ avec $E_1(2)$ est vide, cet automate est 2-localement prédictible.

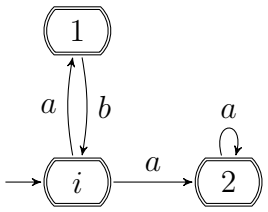


FIGURE 5.3 – Automate 2-localement prédictible

Une particularité de cette propriété est que contrairement au blocs déterminisme, un automate localement prédictible peut être ambigu. Dans

l'exemple précédent, tout mot du langage dénoté par $a(ba)^*$ est l'étiquette de deux chemins réussis. Toutefois, le Théorème 3.2 nous permet de montrer qu'un automate infiniment ambigu ne peut pas être localement prédictible :

Théorème 5.10. *Un automate localement prédictible est finiment ambigu.*

Démonstration. Soient A un automate infiniment ambigu, et p, q, r, s et t cinq états de A. D'après le Théorème 3.2, il existe une lettre a et deux mots u_p et u_s tel qu'il existe deux transitions distinctes (r, a, s) et (r, a, t) dans δ_A, ainsi que les éléments (p, u_p, r), (s, u_s, p), (t, u_s, q) et $(q, u_p a u_s, q)$ dans δ_A^t comme illustré en Figure 5.4. Pour tout entier k strictement positif, l'intersection des ensembles $E_{k-1}(s)$ avec $E_{k-1}(t)$ n'est donc pas vide. Donc A n'est pas localement prédictible. ■

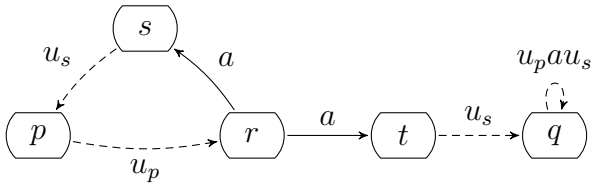


FIGURE 5.4 – Illustration de la preuve du Théorème 5.10

Brzozowski et Santean [11] montrent la décidabilité de la prédictibilité locale d'un automate A pour un k donné, en ré-utilisant un outil pour travailler sur le degré d'ambiguïté d'un automate : l'intersection d'un automate avec lui-même (cf. Définition 3.6 en page 49). D'après la Définition 5.8, un automate A n'est pas k-localement prédictible si et seulement s'il existe un état p possédant au moins deux transitions sortantes distinctes étiquetées par la même lettre a vers des états q_1 et q_2, tel que ces derniers ont au moins une étiquette de longueur $k-1$ en commun. Or, cela est équivalent à l'existence dans l'intersection de A par lui-même, de deux états accessibles (p, p) et (q_1, q_2) et de la transition $((p, p), a, (q_1, q_2))$ tel qu'il existe un chemin de longueur $k-1$ partant de (q_1, q_2).

Lemme 5.11 ([11])**.** *Soient A un automate avec un unique état initial, et A^2 la partie accessible de $A \cap A$. Alors A est k-localement prédictible si et seulement si pour tout état (p, q) de A^2 tel que p et q sont distincts, l'ensemble $E_{k-1}((p, q))$ est vide.*

Remarquons que cela donne également une condition nécessaire et suffisante pour déterminer si un automate est localement prédictible :

Corollaire 5.12 ([11]). *Soient A un automate avec un unique état initial, et A^2 la partie accessible de $A \cap A$. Alors A est localement prédictible si et seulement s'il n'existe pas de cycle non trivial dans A^2 qui soit accessible depuis un état (p, q) tel que p et q soient distincts.*

Ce Corollaire permet de déduire la borne maximale pour la recherche d'un indice de prédictibilité locale, en calculant la plus longue séquence possible d'états de A^2 ne conduisant pas à un cycle non trivial :

Théorème 5.13 ([11]). *Si un automate A est localement prédictible, alors il est au plus $(\binom{n}{2} + 1)$-localement prédictible, avec $n = |Q_A|$, et cette borne peut être atteinte.*

Démonstration. Soient A^2 la partie accessible de $A \cap A$, p et q deux états distincts de A, et w un mot non vide tel que $((p, q), w, (r, s))$ est un élément de $\delta^t_{A^2}$. Les auteurs étudient différents chemins possibles partant de (p, q) :

1. si $(r, s) = (p, q)$, alors (p, q) fait partie d'un cycle non trivial de A^2.
2. si $(r, s) = (q, p)$, alors il existe deux chemins étiquetés par w dans A : un de p vers q et un de q vers p. Donc il existe un chemin étiqueté par w^2 dans A^2 allant de (p, q) vers lui-même.
3. si $(r, s) = (p, p)$, alors p fait partie d'un cycle non trivial de A. Donc (p, p) fait partie d'un cycle non trivial de A^2, accessible depuis (p, q). La même logique vaut dans le cas où $(r, s) = (q, q)$,
4. si $r = s$ et qu'il existe un élément $((r, r), v, (r, t))$ dans δ_{A^2} avec v un mot non vide, alors r fait partie d'un cycle non trivial de A. Donc (r, r) fait partie d'un cycle non trivial de A^2, accessible depuis (p, q).

Donc si A^2 vérifie un de ces points, alors A n'est pas localement prédictible. Supposons maintenant que A soit localement prédictible. Nous essayons de construire la plus longue séquence d'état partant de (p, q). Les points (1) et (2) impliquent que celle-ci ne contient ni doublon, ni couples d'états distincts dont le symétrique est déjà présent, ramenant cette longueur maximale à $\binom{n}{2} + n$. Les points (3) et (4) impliquent que s'il existe un couple (r, r) dans notre séquence, ce couple ne peut être ni précédé ni succédé par les $n-1$ autres couples d'états distincts comprenant r. Afin de maximiser la longueur de la séquence, nous n'incluons donc pas les n couples d'états non distincts. La séquence la plus longue est donc de longueur $\binom{n}{2}$. Si l'on note les états de A par q_1, q_2, …, q_n, cette séquence pourrait être de la forme (q_1, q_2), (q_1, q_3), …, (q_1, q_n), (q_2, q_3), …, (q_2, q_n), …, (q_{n-2}, q_{n-1}), (q_{n-2}, q_n), (q_{n-1}, q_n). Ainsi, $E_{\binom{n}{2}-1}((q_1, q_2))$ n'est pas vide mais $E_{\binom{n}{2}}((q_1, q_2))$ l'est, et A est donc $(\binom{n}{2} + 1)$-localement prédictible.
L'Exemple 5.14 montre alors que cette borne peut être atteinte. ■

Exemple 5.14. Soit A l'automate à quatre états représenté en Figure 5.5. Le non déterminisme de A intervient uniquement au moment de choisir une des deux transitions sortant de i par a. Le plus long chemin dans A^2 partant de $(i,1)$ est le suivant : $((i,1),b,(i,2))$ $((i,2),c,(i,f))$ $((i,f),d,(1,2))$ $((1,2),e,(1,f))$ $((1,f),f,(2,f))$. Donc $E_5((i,1))$ n'est pas vide alors que $E_6((i,1))$ l'est, et A est strictement 7-localement prédictible.

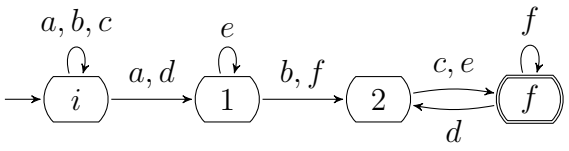


FIGURE 5.5 – Un automate 7-localement prédictif

5.2 La famille des langages localement prédictibles

Une expression rationnelle est k-localement prédictible si son automate des positions est k-localement prédictible. Un langage rationnel est alors dit *k-localement prédictible* s'il peut être dénoté par une expression rationnelle k-localement prédictible, et *localement prédictible* s'il est k-localement prédictible pour un certain k.

Un automate 1-localement prédictible étant un AFD, la famille des langages 1-localement prédictible est la même que celle des langages déterministes (et 1-blocs déterministes). Cette dernière est également strictement incluse dans celle des langages localement prédictibles :

Exemple 5.15. Soient P l'automate des positions de l'expression rationnelle $(b^*a)^*(a+b)$ représenté en Figure 5.6, et M l'AFD minimal du langage dénoté représenté en Figure 5.7.
Remarquons que les états i et a_2 de P ont deux successeurs directs par a et par b, dont les états a_3 et b_4 sans successeur. Donc pour tout mot dans le langage droit de i ou a_2 de longueur supérieure ou égale à 2, la première transition à emprunter est soit celle vers a_2 soit celle vers b_1. D'après la Définition 5.8, P est donc 2-localement prédictible.
Par ailleurs, M étant composé d'une unique orbite non triviale sans transition de synchronisation, il ne valide pas le BW-test et le langage reconnu n'est donc pas déterministe.

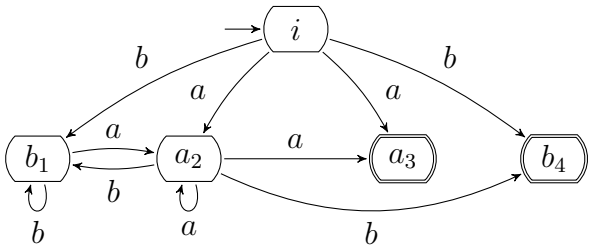


FIGURE 5.6 – L'automate des positions de $(b^*a)^*(a+b)$

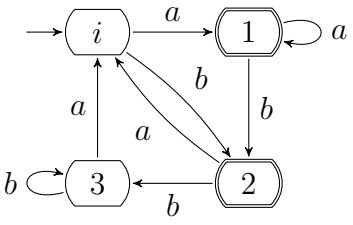


FIGURE 5.7 – L'AFD minimal du langage dénoté par $(b^*a)^*(a+b)$

Nous cherchons alors à nouveau à étendre l'utilisation du BW-test, ici aux automates k-localement prédictibles, pour en déduire une expression rationnelle k-localement prédictible. Pour ce faire, nous définissons une autre relation entre automate et AFD qui préserve et reflète le BW-test.

Soient A et B deux automates, et τ une bijection d'un sous-ensemble de $\Sigma_A \times Q_A$ vers Σ_B. Nous disons que B est une *image transitionnelle de A par τ* , notée $\tau(A)$, si $Q_B = Q_A$, $I_B = I_A$, $F_B = F_A$ et $\delta_B = \{(p, \tau(a,q), q) \mid (p,a,q) \in \delta_A\}$. Cette relation, tout comme l'image alphabétique, préserve et reflète la finalité des états, les équivalences d'étiquettes des transitions arrivant dans un même état, et donc les propriétés structurelles tel que l'homogénéité, la forte stabilité et la forte transversalité. L'image transitionnelle d'un automate est donc un automate des positions (respectivement valide le BW-test) si et seulement si cet automate est un automate des positions (respectivement valide le BW-test). Remarquons également que puisque τ est une bijection, alors B est nécessairement déterministe. Ce qui nous permet d'étendre le Théorème 3.51 également aux langages k-localement prédictibles :

Théorème 5.16. *Un langage est k-localement prédictible si et seulement s'il est reconnaissable par un automate k-localement prédictible et validant le BW-test.*

Démonstration. Soit L un langage rationnel.

Si L est k-localement prédictible, alors il est reconnu par un automate des positions k-localement prédictible. Le Théorème 3.50 permet de conclure.

Supposons maintenant que L soit reconnaissable par un automate A tel que A soit k-localement prédictible et valide le BW-test. Soient $\Pi = \{a_q \mid \exists(p, a, q) \in \delta_A\}$ un alphabet, τ une bijection d'un sous-ensemble de $\Sigma_A \times Q_A$ vers Π, et B un automate tel que $B = \tau(A)$. Comme dit plus haut, B est déterministe et valide le BW-test. Donc d'après le Théorème 3.51, $\mathrm{L}(B)$ est déterministe et il existe un automate des positions déterministe P reconnaissant $\mathrm{L}(B)$. Nous définissons alors le morphisme θ (similaire à $\natural$) de Π vers Σ_A tel que $\theta(a_q) = a$ et l'étendons à un morphisme de Π^* vers Σ_A^*. En appliquant alors le morphisme θ sur les étiquettes des transitions de P, nous obtenons un automate G. Comme P est un automate des positions, G l'est également.
Nous étudions alors les langages droits de G. D'après la définition de θ, nous avons $\mathrm{L}(G) = \theta(\mathrm{L}(P)) = \theta(\mathrm{L}(B)) = \mathrm{L}(A) = L$. Plus généralement, pour tout état q de A, nous avons $\mathrm{L}_A(q) = \theta(\mathrm{L}_B(q))$; et pour tout état q' de G, nous avons $\mathrm{L}_G(q') = \theta(\mathrm{L}_P(q'))$.
Il reste alors à montrer que G est k-localement prédictible. Ce dernier étant un automate des positions, il n'a donc qu'un seul état initial. Soient $(p', \theta(a_q), q')$ et $(p', \theta(a_r), r')$ deux transitions distinctes de G (étiquetées donc par a). Par définition d'une image transitionnelle, (p', a_q, q') et (p', a_r, r') sont deux transitions distinctes de P. Puisque P et B sont deux AFD équivalents, alors il existe dans B deux transitions distinctes (p, a_q, q) et (p, a_r, r) tel que p, q et r sont trois états de B respectivement équivalents aux états p', q' et r' de P. Encore par définition d'une image transitionnelle, (p, a, q) et (p, a, r) sont deux transitions distinctes de A. Comme A est k-localement prédictible, alors l'intersection de $E_{k-1}(q)$ avec $E_{k-1}(r)$ est vide dans A. Or, puisque q et r dans B sont respectivement équivalents à q' et r' dans P, alors $\mathrm{L}_G(q') = \mathrm{L}_A(q)$ et $\mathrm{L}_G(r') = \mathrm{L}_A(r)$. L'intersection de $E_{k-1}(q')$ avec $E_{k-1}(r')$ dans G est donc également vide. Donc G est un automate des positions k-localement prédictible reconnaissant L, et L est bien k-localement prédictible. ■

Dans la suite de ce Chapitre, nous étudions la hiérarchie interne de cette famille et ses relations avec la famille des langages blocs déterministes.

5.3 Hiérarchie

Dans cette Section, nous démontrons qu'il existe une hiérarchie interne infinie au sein de la famille des langages localement prédictibles. Pour cela, nous définissons la famille de langages paramétrée suivante :

Définition 5.17. Soit n un entier naturel. L'automate unaire $A_n = (\{a\}, Q, I, F, \delta)$ est défini de la manière suivante :

- $Q = \{\alpha_i \mid 0 \leq i \leq 2n\}$,
- $I = \{\alpha_0\}$,
- $F = \{\alpha_0, \alpha_n\}$,
- $\delta = \{(\alpha_i, a, \alpha_{i+1}) \mid 0 \leq i < 2n\} \cup \{(\alpha_{2n}, a, \alpha_0)\}$.

La famille d'automates unaires $\mathcal{A}$ est alors définie par $\{A_j \mid j \in \mathbb{N} \setminus \{0\}\}$.

Un représentant de cette famille est illustré en Figure 5.8 :

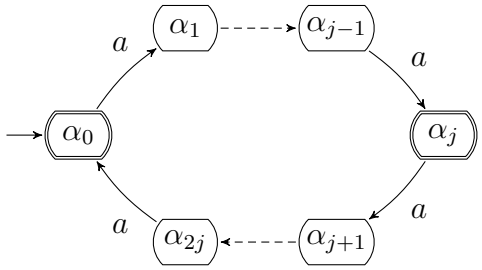


FIGURE 5.8 – Un automate A_j de la famille $\mathcal{A}$

Nous observons qu'aucun automate A_j ne valide le BW-test car il n'existe pas de transition de synchronisation. Toutefois, il est possible d'effectuer l'opération représentée en Figure 5.9, consistant à partitionner en deux les langages droits des états α_1 à α_j, de sorte qu'ils sont représentés par l'union disjointe des langages droits des états β et γ de mêmes indices. Le langage reconnu est ainsi préservé et, comme les langages droits des états γ sont finis, la prédictibilité locale également. Par ailleurs, le BW-test est validé car l'unique orbite non triviale n'a plus qu'une seule porte de sortie.

Proposition 5.18. *Le langage reconnu par un automate A_j de $\mathcal{A}$ est $(j+1)$-localement prédictible et peut être dénoté par l'expression $(a^{2j+1})^* \cdot (\varepsilon + a^j)$.*

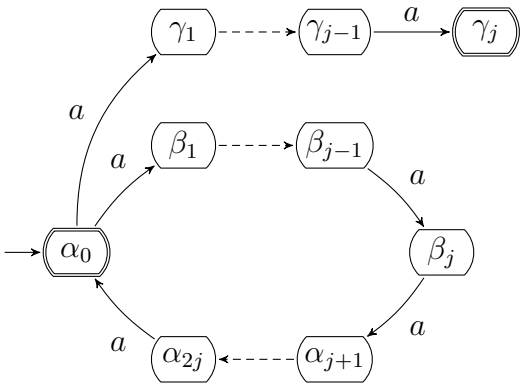


FIGURE 5.9 – L'automate A'_j validant le BW-test

Il nous faut alors montrer que ce langage n'est pas j-localement prédictible. Tout d'abord, nous montrons que les langages reconnus par deux automates distincts dans la famille $\mathcal{A}$ sont distincts :

Proposition 5.19. *Tout automate A_j de $\mathcal{A}$ est un AFD minimal.*

Démonstration. Par construction, tout automate A_j de $\mathcal{A}$ est émondé et déterministe. Sinon, il existe deux états finaux α_0 et α_j tel que a^j appartient au langage droit de α_0 et pas à celui de α_j. Ces deux états ne sont donc pas équivalents, et les états non-finaux n'ont pas d'équivalents distincts. Donc tout automate A_j de $\mathcal{A}$ est un AFD minimal. ■

Nous démontrons alors qu'il n'existe pas d'automate des positions j-localement prédictible équivalent à A_j. Pour cela, nous utilisons une propriété simple des automates localement prédictibles sur un alphabet unaire :

Proposition 5.20. *Soit A un automate localement prédictible sur un alphabet unaire. Alors pour tout état de A, au plus un de ses successeurs directs a un langage droit infini.*

Démonstration. Soit a la lettre de l'alphabet de A. Soient p et q deux états distincts de A, successeurs directs d'un même état. Si le langage droit d'un état est infini, alors son ensemble d'étiquettes contient tous les mots non vide sur $\{a\}$. Donc si p et q ont un langage droit infini, alors pour tout entier positif k, l'intersection de $E_k(p)$ avec $E_k(q)$ n'est pas vide, contredisant alors le fait que A soit localement prédictible. ■

La caractérisation des automates des positions en Section 2.3 permet alors de déduire la propriété suivante sur la structure orbitale des automates des positions sur un alphabet unaire et localement prédictible :

Proposition 5.21. *Soit P un automate des positions sur un alphabet unaire, qui est localement prédictible et reconnaît un langage infini. Alors P a exactement une orbite non-triviale O, avec une seule porte d'entrée et une seule porte de sortie, et tel que parmi les successeurs directs de n'importe lequel de ses états, un seul soit dans O.*

Démonstration. D'après la Proposition 5.20, il ne peut exister d'embranchement pouvant mener à plusieurs orbites non triviales distinctes. Puisque P a un unique état initial et reconnaît un langage infini, il contient une unique orbite non triviale O, accessible par une unique porte d'entrée g_i.

Supposons alors qu'il existe un état de O avec deux successeurs directs dans O : ces successeurs ayant tous deux un langage droit infini, cela contredirait la Proposition 5.20.

Supposons enfin que O ait deux portes de sortie g_{o1} et g_{o2} distinctes. D'après le Théorème 2.28 (en page 46), O est stable et g_i serait alors un successeur direct de g_{o1} et de g_{o2}. Il existerait donc un état de O avec deux successeurs directs dans O. ■

Remarquons alors que d'après la Proposition 5.20, il ne peut exister d'état avec deux successeurs directs distincts pouvant accéder à la porte d'entrée de l'orbite non triviale O. Il existe donc un unique chemin allant de l'état initial jusqu'à sa porte d'entrée sans traverser d'état de O. Le schéma en Figure 5.10 résume les propriétés énoncées concernant un automate des positions localement prédictible sur un alphabet unaire :

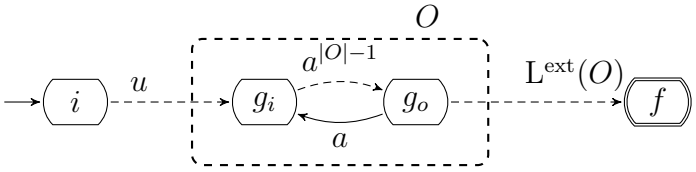


FIGURE 5.10 – Résumé des propriétés énoncées

Nous terminons par une propriété du langage reconnu par un automate A_j de $\mathcal{A}$. Ce dernier étant unaire, il est possible d'ordonner totalement ses mots par rapport à leur longueur : $\{\varepsilon, a^j, a^{2j+1}, a^{3j+1}, a^{4j+2}, \ldots\}$. Nous obtenons alors que :

Lemme 5.22. *Soient A_j un automate de $\mathcal{A}$, et w_1, w_2 et w_3 trois mots consécutifs de $L(A_j)$. Alors, soit $|w_3| - |w_2| = j + 1$ et $|w_2| - |w_1| = j$, ou $|w_3| - |w_2| = j$ et $|w_2| - |w_1| = j + 1$.*

Ce qui nous permet d'achever notre démonstration :

Proposition 5.23. *Le langage reconnu par un automate A_j de $\mathcal{A}$ n'est pas j-localement prédictible.*

Démonstration. Soit P un automate des positions j-localement prédictible reconnaissant un langage unaire infini. D'après la Proposition 5.21, il existe une unique orbite non triviale O dans P, avec une unique porte d'entrée g_i et une unique porte de sortie g_o. Montrons que P ne reconnaît pas $\mathrm{L}(A_j)$.

Nous étudions d'abord la taille des mots étiquetant les chemins allant de l'état initial i de P vers g_o. D'après la Proposition 5.21, chaque état o de O a exactement un successeur direct qui est dans O. Donc tout chemin d'un état de o vers lui même est étiqueté par un mot de longueur égale à $k \times |O|$ avec k un entier naturel. Puisque les orbites d'un automate des positions sont stables, alors g_i est un successeur direct de g_o, et tout chemin de g_i vers g_o est étiqueté par un mot de longueur égale à $(k+1) \times |O| - 1$ avec k un entier naturel. Soit m la longueur du mot u étiquetant l'unique chemin allant de i vers g_i sans traverser d'état de O. Les chemins allant de i vers g_o sont donc étiquetés par des mots de longueur égale à $m - 1 + (k+1) \times |O|$ avec k un entier naturel : la différence de longueur entre deux mots consécutifs est donc constante.

Nous terminons par l'étude du langage externe de O. Puisque P est j-localement prédictif, la longueur des mots de $\mathrm{L}^{\mathrm{ext}}(O)$ est strictement inférieure à j. Trois cas se présentent alors.
Soit $\mathrm{L}^{\mathrm{ext}}(O)$ est vide et $\mathrm{L}(P)$ est fini, et donc différent de $\mathrm{L}(A_j)$.
Soit il existe au moins deux mots v_1 et v_2 dans $\mathrm{L}^{\mathrm{ext}}(O)$, tel que $|v_2| > |v_1|$. Donc pour tout mot w étiquetant les chemins allant de i vers g_o, les mots wv_1 et wv_2 appartiennent à $\mathrm{L}(P)$ tel que $|wv_2| - |wv_1| = |v_2| - |v_1| < j$. Il existe donc deux mots consécutifs du langage dont la différence de longueur est strictement inférieure à j, ce qui, d'après le Lemme 5.22, implique que P ne reconnaît pas $\mathrm{L}(A_j)$.
Soit $\mathrm{L}^{\mathrm{ext}}(O)$ contient un unique mot w_o de longueur m'. Puisque O est la seule orbite non triviale et a une unique porte de sortie, il existe un entier n tel que pour tout mot w dans $\mathrm{L}(P)$ de longueur supérieure à n, il existe w_p l'étiquette d'un chemin allant de i vers g_o, tel que $w = w_p w_o$. D'après ce que nous avons vu au paragraphe précédent, nous obtenons que la longueur de w est égale à $m - 1 + m' + (k+1) \times |O|$. La différence de longueur entre tout deux mots consécutifs est donc égale à $|O|$, c'est-à-dire une constante. Ce qui, d'après le Lemme 5.22, implique que P ne reconnaît pas $\mathrm{L}(A_j)$. ■

Ce qui nous permet de conclure :

Théorème 5.24. *Il existe une hiérarchie infinie de familles de langages strictement j-localement prédictibles.*

5.4 Relation d'inclusion avec les bloc-déterministes

Le fait que pour tout entier positif k, les langages k-blocs déterministes forment une sous-famille des langages k-localement prédictibles peut être déduit directement de la construction de l'automate des positions. Soient E_b une expression rationnelle k-blocs déterministes, et E l'expression rationnelle construite à partir de E_b en remplaçant chaque bloc $[b_1 \cdots b_n]$ par l'expression $b_1 \cdots b_n$. Alors E dénote le même langage que E_b.

Soit x une position de $E^\sharp$. Alors $|\text{Follow}(E^\sharp, x)| > 1$ si et seulement si $x^\natural$ est la dernière lettre d'un bloc de E_b. Donc chaque position distincte succédant directement à x dans E est la première lettre d'un bloc (distinct) de E_b. Comme E_b est k-blocs et déterministe, alors toutes les positions distinctes succédant directement à x sont celles de blocs de taille au plus k qui ne sont pas préfixes les uns des autres. Donc pour toute position de $E^\sharp$, la lecture des k prochaines lettres en entrée détermine totalement la prochaine position. Ce qui implique que l'automate des positions de E est k-localement prédictible. Une illustration est donnée ci-dessous :

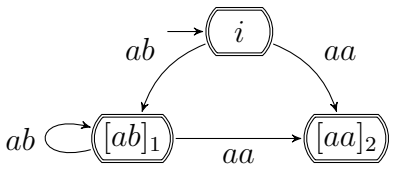


FIGURE 5.11 – $E_b = [ab]^*(\varepsilon + [aa])$

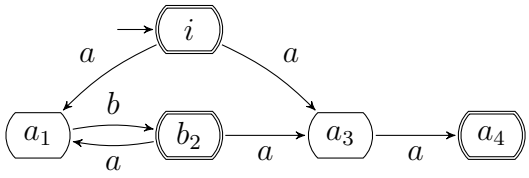


FIGURE 5.12 – $E = (ab)^*(\varepsilon + aa)$

Han et Wood [36] affirment que la famille des langages blocs déterministes est une sous-famille stricte des langages localement prédictibles. Leur preuve se base sur une affirmation de Giammarresi *et al.* [31] comme quoi les langages de la famille $\text{L}((b^*a)^*(a+b)^k)$ ne sont pas blocs déterministes pour tout entier k. Or, nous avons invalidé la Conjecture 1 qu'ils utilisent comme argument, et donnons donc notre propre preuve.

Nous avons montré que pour tout automate A_j de $\mathcal{A}$, le langage $\text{L}(A_j)$ est strictement $(j+1)$-localement prédictible. Donc, pour tout entier j strictement positif, ce langage n'est pas déterministe et, selon le Théorème 4.50, n'est pas blocs déterministe.

Proposition 5.25. *Pour tout entier k strictement positif, il existe des langages rationnels unaires qui sont k-localement prédictibles sans être blocs déterministes.*

Ce qui nous permet de conclure par ces deux propriétés :

Théorème 5.26. *1. La famille des langages blocs déterministes est strictement incluse dans celle des langages localement prédictibles.*

2. Pour tout entier k strictement positif, la famille des langages k-blocs déterministes est strictement incluse dans celle des langages k-localement prédictibles.

Pour terminer, nous montrons que pour tout entier k strictement positif, il existe des langages strictement k-blocs déterministes qui sont également 2-localement prédictibles. Pour certains langages, il est donc possible de réduire la taille de la fenêtre de symboles à lire sur l'entrée.

Définition 5.27. Soit k un entier supérieur ou égal à 2. L'automate $C_k = (\{a, b, c\}, Q, I, F, \delta)$ est défini de la manière suivante :

— $Q = \{i, f\} \cup \{\alpha_j, \beta_j \mid 1 \leq j \leq k-1\}$,

— $I = \{i\}$,

— $F = \{i, f\}$,

— $\delta = \{(i, a, f), (i, b, \beta_1), (f, a, \alpha_1), (\alpha_{k-1}, a, i), (\beta_{k-1}, a, i), (\beta_{k-1}, c, i)\}$
$\cup \{(\alpha_j, b, \alpha_{j+1}), (\beta_j, b, \beta_{j+1}) \mid 1 \leq j \leq k-2\}$.

La famille d'automates $\mathcal{C}$ est alors définie par $\{C_k \mid k \in \mathbb{N} \setminus \{0, 1\}\}$.

Un représentant de cette famille est illustré en Figure 5.13 :

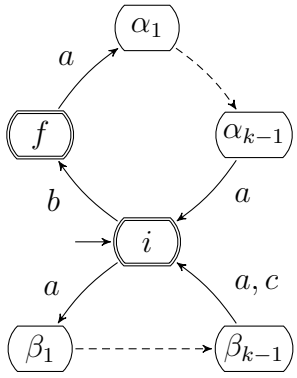


FIGURE 5.13 – Un automate C_k de la famille $\mathcal{C}$

Tous les automates de la famille $\mathcal{C}$ sont des AFD minimaux reconnaissant des langages distincts. Nous remarquons qu'il n'existe pas de transition de synchronisation et que le BW-test n'est donc pas validé. Il est toutefois possible d'obtenir l'automate k-blocs et déterministe validant le BW-test,

représenté en Figure 5.14 : il est alors nécessaire d'étendre les transitions autre que celle entre i et f afin qu'il ne reste plus que ces deux états. Un automate C_k reconnaît donc le langage strictement k-blocs déterministe dénoté par $(\varepsilon + [a^{k-1}c]^*b)([a^k](\varepsilon + [a^{k-1}c]^*b)^*$. Mais il est également possible d'obtenir l'automate 2-localement prédictible validant le BW-test représenté en Figure 5.15, en partitionnant le langage droit de l'état f afin de le rendre non final et de faire de l'état i l'unique porte de sortie de son orbite. Nous obtenons alors que le langage reconnu peut être dénoté par l'expression rationnelle 2-localement prédictible $(ba^k + a^{k-1}(a+c))^*(\varepsilon + b)$.

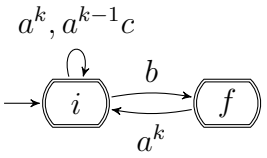


FIGURE 5.14 – ABD k-blocs validant le BW-test

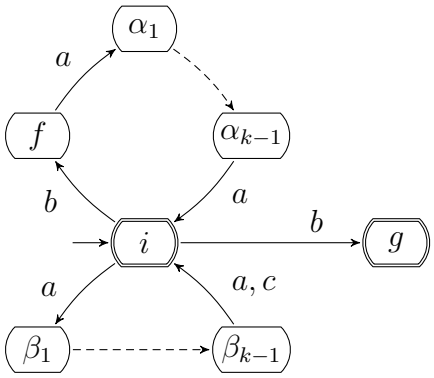


FIGURE 5.15 – Automate 2-localement prédictible validant le BW-test

Théorème 5.28. *Pour tout entier k supérieur ou égal à 3, il existe des langages strictement k-blocs déterministes qui sont 2-localement prédictibles.*

5.5 Ouverture

Caron *et al.* [18] ont également présenté une extension à la prédictibilité locale dans les automates : la (k,l)-*non ambiguïté*. Contrairement aux automates k-localement prédictibles, les états peuvent avoir plusieurs successeurs directs ayant des étiquettes de longueur $k-1$ en commun, tant qu'il existe un successeur commun à ces états par un préfixe de l'étiquette commune. Nous pourrions alors définir une nouvelle famille de langages reconnaissables par des automates des positions (k,l)-non ambigus.

Cette propriété sur un automate permet aux auteurs de construire ce qu'ils appellent *une structure quasi-déterministe*, indépendante de l'automate de départ et permettant de reconnaître son langage. Cette structure est similaire à un automate acyclique auquel est ajouté des transitions éti-

quetées d'entiers strictement positifs précisant de combien de lettres la fenêtre de lecture du mot en entrée doit être décalée. Les auteurs montrent notamment que l'appartenance d'un mot au langage se fait en temps linéaire sur la taille du mot, et que pour certains langages, il existe une structure quasi-déterministe dont la taille est exponentiellement plus petite que celle de leur AFD minimal. Cette structure quasi-déterministe pourrait alors être vue comme une forme compacte d'un délégateur, autorisant des décalages de taille variable.

Chapitre 6

Conclusion

6.1 Bilan

Au cours de cette thèse, nous avons été amenés à avoir un regard critique sur les résultats énoncés dans les études précédentes, qui étaient par ailleurs les seules dans ce domaine. La décidabilité du blocs déterminisme d'un langage avait en effet été énoncée sans jamais en avoir apporté une preuve complète. Nous avons pu réfuter cette affirmation grâce à une étude des propriétés structurelles des automates à blocs déterministes et de leur relation avec les AFD minimaux. S'en est directement suivie la réfutation de la preuve de la hiérarchie propre de la famille des langages blocs déterministes. Si cette hiérarchie a pu être redémontrée, la décidabilité du blocs déterminisme d'un langage rationnel n'a été prouvée que dans le cas des unaires, prouvant par la même occasion que la famille des langages blocs déterministes est strictement incluse dans celle des rationnels. Dans le cas d'un langage rationnel non unaire, nous avons pu prouver la décidabilité pour une taille maximale de blocs fixée, mais il nous manque encore l'existence d'une borne pour la décidabilité de l'appartenance de ce langage dans la famille des langages blocs déterministes.

Concernant la famille des langages localement prédictibles, nous avons essentiellement étudié les propriétés des langages rationnels unaires. En combinant les propriétés des automates des positions, des automates localement prédictibles et celles des langages unaires, nous avons pu démontrer l'existence d'une hiérarchie propre au sein de la famille. Ces résultats nous ont également permis de démontrer que la famille des langages blocs déterministes est strictement incluse dans celle des langages localement prédictibles. Les questions de décidabilité de la prédictibilité locale, voir simplement de la k-prédictibilité locale restent ouvertes.

6.2 Perspectives

D'autres familles de langages peuvent être définies, comme celle des (k, l)-non-ambigus. Mais il est aussi possible d'étendre celles déjà existantes.

Caron *et al.* [15, 16] ont étudié les conditions nécessaires pour qu'un opérateur dans une expression rationnelle soit compatible avec la construction de l'automate des positions. En effet, certains opérateurs rationnels, utilisés au sein d'une expression rationnelle étendue, ne vérifient pas la compatibilité avec la linéarisation :

— dans le cas de l'intersection : si $E = a \cap a$ alors $E^\sharp = a_1 \cap a_2$, et $(\mathrm{L}(E^\sharp))^\natural = (\{a_1\} \cap \{a_2\})^\natural = \emptyset$ alors que $\mathrm{L}(E) = \{a\}$,

— ou dans le cas du complémentaire : si $E = \neg a + \neg a$ alors $E^\sharp = \neg a_1 + \neg a_2$, et $(\mathrm{L}(E^\sharp))^\natural = (\Pi_E^* \setminus \{a_1\} \cup \Pi_E^* \setminus \{a_2\})^\natural = (\Pi_E^*)^\natural = \Sigma^*$ alors que $\mathrm{L}(E) = \Sigma^* \setminus \{a\}$.

Ils introduisent également de nouveaux opérateurs compatibles avec cette construction, consistant à ajouter ou supprimer le mot vide du langage dénoté par certaines sous-expressions. Ces derniers offrent notamment la possibilité d'obtenir des expressions rationnelles plus compactes.

Pour les opérateurs compatibles avec la construction de l'automate des positions, il existe également la construction de l'automate des Follows d'Illie et Yu [39] et celle de l'automate des dérivées partielles d'Antimirov [2, 22] dont il a été montré que ce sont des quotients de l'automate des positions. La question serait notamment de savoir s'il existe des langages non déterministes pouvant être reconnus par un automate déterministe construit selon ces méthodes.

Concernant les opérateurs non compatibles avec la construction de l'automate des positions, certains auteurs proposent de nouvelles constructions comme Caron *et al.* dont nous avons parlé en Section 3.3. D'autres proposent de généraliser la construction de l'automate des positions, comme Broda *et al.* [7] pour l'intersection, qui utilisent des ensembles de positions pour représenter les états au lieu de simples positions.

Bibliographie

[1] Cyril Allauzen, Mehryar Mohri et Ashish Rastogi. « General Algorithms for Testing the Ambiguity of Finite Automata and the Double-Tape Ambiguity of Finite-State Transducers ». In : *Int. J. Found. Comput. Sci.* 22.4 (2011), p. 883-904 (cf. p. 50).

[2] Valentin M. Antimirov. « Partial Derivatives of Regular Expressions and Finite Automaton Constructions ». In : *Theor. Comput. Sci.* 155.2 (1996), p. 291-319 (cf. p. 138).

[3] Dean N. Arden. « Delayed-logic and finite-state machines ». In : *2nd Annual Symposium on Switching Circuit Theory and Logical Design, Detroit, Michigan, USA, October 17-20, 1961*. IEEE Computer Society, 1961, p. 133-151 (cf. p. 63).

[4] Jean Berstel et Jean-Eric Pin. « Local Languages and the Berry-Sethi Algorithm ». In : *Theor. Comput. Sci.* 155.2 (1996), p. 439-446 (cf. p. 13).

[5] Ronald Vernon Book et al. « Ambiguity in Graphs and Expressions ». In : *IEEE Trans. Computers* 20.2 (1971), p. 149-153 (cf. p. 2, 50-52).

[6] Tim Bray et al. *Extensible markup language (XML) 1.0*. 2008 (cf. p. 2).

[7] Sabine Broda et al. « Position Automaton Construction for Regular Expressions with Intersection ». In : *Developments in Language Theory - 20th International Conference, DLT 2016, Montréal, Canada, July 25-28, 2016, Proceedings*. Sous la dir. de Srecko Brlek et Christophe Reutenauer. T. 9840. Lecture Notes in Computer Science. Springer, 2016, p. 51-63 (cf. p. 138).

[8] J. A. Brzozowski. « Canonical regular expressions and minimal state graphs for definite events ». In : *Mathematical Theory of Automata*. Volume 12 of MRI Symposia Series. Polytechnic Press, Polytechnic Institute of Brooklyn, N.Y., 1962, p. 529-561 (cf. p. 27).

[9] Janusz A. Brzozowski. « Derivatives of Regular Expressions ». In : *J. ACM* 11.4 (1964), p. 481-494 (cf. p. 20, 22).

[10] Janusz A. Brzozowski et Edward J. McCluskey. « Signal Flow Graph Techniques for Sequential Circuit State Diagrams ». In : *IEEE Trans. Electronic Computers* 12.2 (1963), p. 67-76 (cf. p. 15).

[11] Janusz A. Brzozowski et Nicolae Santean. « Predictable semiautomata ». In : *Theor. Comput. Sci.* 410.35 (2009), p. 3236-3249 (cf. p. 124, 125).

[12] Janusz A. Brzozowski et Hellis Tamm. « Theory of átomata ». In : *Theor. Comput. Sci.* 539 (2014), p. 13-27 (cf. p. 27).

[13] Anne Brüggemann-Klein. « Regular Expressions into Finite Automata ». In : *Theor. Comput. Sci.* 120.2 (1993), p. 197-213 (cf. p. 32, 34, 35, 55).

[14] Anne Brüggemann-Klein et Derick Wood. « One-Unambiguous Regular Languages ». In : *Inf. Comput.* 140.2 (1998), p. 229-253 (cf. p. 2, 39, 53, 54, 56, 57, 61, 66, 70, 72, 74, 77, 119).

[15] Pascal Caron, Jean-Marc Champarnaud et Ludovic Mignot. « Multi-Bar and Multi-Tilde Regular Operators ». In : *Journal of Automata, Languages and Combinatorics* 16.1 (2011), p. 11-26 (cf. p. 138).

[16] Pascal Caron, Jean-Marc Champarnaud et Ludovic Mignot. « Multi-tilde-bar expressions and their automata ». In : *Acta Inf.* 49.6 (2012), p. 413-436 (cf. p. 138).

[17] Pascal Caron et Marianne Flouret. « On Glushkov K-graphs ». In : *Scientific Applications of Language Methods.* Sous la dir. de Carlos Martín-Vide. T. 2. Mathematics, Computing, Language, and Life : Frontiers in Mathematical Linguistics and Language Theory. Imperial College Press, 2010, p. 103-132 (cf. p. 36).

[18] Pascal Caron, Marianne Flouret et Ludovic Mignot. « (k, l)-Unambiguity and Quasi-Deterministic Structures : An Alternative for the Determinization ». In : *Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings.* Sous la dir. d'Adrian-Horia Dediu et al. T. 8370. Lecture Notes in Computer Science. Springer, 2014, p. 260-272 (cf. p. 3, 135).

[19] Pascal Caron, Yo-Sub Han et Ludovic Mignot. « Generalized One-Unambiguity ». In : *Developments in Language Theory - 15th International Conference, DLT 2011, Milan, Italy, July 19-22, 2011. Proceedings*. Sous la dir. de Giancarlo Mauri et Alberto Leporati. T. 6795. Lecture Notes in Computer Science. Springer, 2011, p. 129-140 (cf. p. 78).

[20] Pascal Caron et Djelloul Ziadi. « Characterization of Glushkov automata ». In : *Theor. Comput. Sci.* 233.1-2 (2000), p. 75-90 (cf. p. 4, 36, 39).

[21] J-M Champarnaud, J-L Ponty et Djelloul Ziadi. « From regular expressions to finite automata ». In : *International journal of computer mathematics* 72.4 (1999), p. 415-431 (cf. p. 78).

[22] Jean-Marc Champarnaud et Djelloul Ziadi. « From C-Continuations to New Quadratic Algorithms for Automaton Synthesis ». In : *IJAC* 11.6 (2001), p. 707-736 (cf. p. 138).

[23] Chia-Hsiang Chang et Robert Paige. « From Regular Expressions to DFA's Using Compressed NFA's ». In : *Theor. Comput. Sci.* 178.1-2 (1997), p. 1-36 (cf. p. 35).

[24] Haiming Chen et Lei Chen. « Inclusion Test Algorithms for One-Unambiguous Regular Expressions ». In : *Theoretical Aspects of Computing - ICTAC 2008, 5th International Colloquium, Istanbul, Turkey, September 1-3, 2008. Proceedings*. Sous la dir. de John S. Fitzgerald, Anne Elisabeth Haxthausen et Hüsnü Yenigün. T. 5160. Lecture Notes in Computer Science. Springer, 2008, p. 96-110 (cf. p. 56).

[25] Noam Chomsky. « Three models for the description of language ». In : *IRE Trans. Information Theory* 2.3 (1956), p. 113-124 (cf. p. 1, 7).

[26] Zhe Dang, Oscar H. Ibarra et Jianwen Su. « Composability of Infinite-State Activity Automata ». In : *Algorithms and Computation, 15th International Symposium, ISAAC 2004, Hong Kong, China, December 20-22, 2004, Proceedings*. Sous la dir. de Rudolf Fleischer et Gerhard Trippen. T. 3341. Lecture Notes in Computer Science. Springer, 2004, p. 377-388 (cf. p. 119).

[27] François Denis, Aurélien Lemay et Alain Terlutte. « Residual Finite State Automata ». In : *Fundam. Inform.* 51.4 (2002), p. 339-368 (cf. p. 20).

[28] Samuel EILENBERG. *Automata, languages, and machines. A*. T. 59 A. Pure and applied mathematics. Academic Press, 1974. ISBN : 0122340019 (cf. p. 79).

[29] Cagdas Evren GEREDE et al. « Automated composition of e-services : lookaheads ». In : *Service-Oriented Computing - ICSOC 2004, Second International Conference, New York, NY, USA, November 15-19, 2004, Proceedings*. Sous la dir. de Marco AIELLO et al. ACM, 2004, p. 252-262 (cf. p. 119).

[30] Dora GIAMMARRESI et Rosa MONTALBANO. « Deterministic generalized automata ». In : *Theoretical Computer Science* 215.1–2 (1999), p. 191 -208. ISSN : 0304-3975 (cf. p. 79, 87, 90, 91, 93, 101).

[31] Dora GIAMMARRESI, Rosa MONTALBANO et Derick WOOD. « Block-Deterministic Regular Languages ». In : *Theoretical Computer Science, 7th Italian Conference, ICTCS 2001, Torino, Italy, October 4-6, 2001, Proceedings*. 2001, p. 184-196 (cf. p. 2, 3, 79, 99, 133).

[32] V. M. GLUSHKOV. « The abstract theory of automata ». In : *Russian Mathematical Surveys* 16 (1961), p. 1-53 (cf. p. 2, 12).

[33] Benoît GROZ et Sebastian MANETH. « Efficient testing and matching of deterministic regular expressions ». In : *J. Comput. Syst. Sci.* 89 (2017), p. 372-399 (cf. p. 55).

[34] Hermann GRUBER et Markus HOLZER. « Provably Shorter Regular Expressions from Finite Automata ». In : *Int. J. Found. Comput. Sci.* 24.8 (2013), p. 1255-1280 (cf. p. 18).

[35] Yo-Sub HAN. « State Elimination Heuristics for Short Regular Expressions ». In : *Fundam. Inform.* 128.4 (2013), p. 445-462 (cf. p. 18).

[36] Yo-Sub HAN et Derick WOOD. « Generalizations of 1-deterministic regular languages ». In : *Inf. Comput.* 206.9-10 (2008), p. 1117-1125 (cf. p. 3, 99, 119, 122, 133).

[37] John. E. HOPCROFT. « An n log n Algorithm for Minimizing the States in a Finite Automaton ». In : *The theory of machines and computations*. Sous la dir. de Z. KOHAVI. New York : Academic Press, 1971, p. 189-196 (cf. p. 27).

[38] John E. HOPCROFT, Rajeev MOTWANI et Jeffrey D. ULLMAN. *Introduction to automata theory, languages, and computation - international edition (2. ed)*. Addison-Wesley, 2003. ISBN : 978-0-321-21029-6 (cf. p. 7, 22-24).

[39] L. ILIE et S. YU. « Follow automata. » In : *Inf. Comput.* 186.1 (2003), p. 140-162 (cf. p. 138).

[40] ISO8879 :1986. *Information processing – Text and Office Systems – Standard Generalized Markup Language (SGML)*. Standard No. ISO 8879 :1986. International Organization for Standardization, 1986 (cf. p. 2, 53).

[41] S. Kleene. « Representation of events in nerve nets and finite automata ». In : *Automata Studies* Ann. Math. Studies 34 (1956). Princeton U. Press, p. 3-41 (cf. p. 2, 12).

[42] A. Layman et al. *XML-Data*. 1998. url : `http://www.w3.org/TR/1998/NOTE-XML-data` (cf. p. 2).

[43] Christof Löding et Stefan Repke. « Decidability Results on the Existence of Lookahead Delegators for NFA ». In : *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2013, December 12-14, 2013, Guwahati, India*. Sous la dir. d'Anil Seth et Nisheeth K. Vishnoi. T. 24. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013, p. 327-338 (cf. p. 120-122).

[44] Katja Losemann, Wim Martens et Matthias Niewerth. « Closure properties and descriptional complexity of deterministic regular expressions ». In : *Theor. Comput. Sci.* 627 (2016), p. 54-70 (cf. p. 70).

[45] Robert McNaughton et Hisao Yamada. « Regular Expressions and State Graphs for Automata ». In : *IRE Trans. Electronic Computers* 9.1 (1960), p. 39-47 (cf. p. 2, 12, 15).

[46] Ludovic Mignot. « Des Codes Barres pour les Langages Rationnels. (Bar Codes for Regular Languages) ». Thèse de doct. University of Rouen, France, 2010 (cf. p. 13, 30).

[47] Edward F Moore. « Gedanken-experiments on sequential machines ». In : *Automata studies* 34 (1956), p. 129-153 (cf. p. 27).

[48] J. Myhill. « Finite automata and the representation of events ». In : *WADD* TR-57-624 (1957), p. 112-137 (cf. p. 19).

[49] Anil Nerode. « Linear automaton transformations ». In : *Proceedings of the American Mathematical Society* 9.4 (1958), p. 541-544 (cf. p. 19, 26).

[50] Jean-Luc Ponty, Djelloul Ziadi et Jean-Marc Champarnaud. « A New Quadratic Algorithm to Convert a Regular Expression into an Automaton ». In : *Workshop on Implementing Automata*. T. 1260. Lecture Notes in Computer Science. Springer, 1996, p. 109-119 (cf. p. 35).

[51] Bala Ravikumar et Nicolae Santean. « On the Existence of Lookahead Delegators for NFA ». In : *Int. J. Found. Comput. Sci.* 18.5 (2007), p. 949-973 (cf. p. 120).

[52] Jacques Sakarovitch. « Automata and rational expressions ». In : *CoRR* abs/1502.03573 (2015). arXiv : 1502.03573. URL : http://arxiv.org/abs/1502.03573 (cf. p. 35).

[53] Jacques Sakarovitch. *Éléments de théorie des automates*. Paris : Vuibert, 2003 (cf. p. 22).

[54] Robert Endre Tarjan. « Depth-First Search and Linear Graph Algorithms ». In : *SIAM J. Comput.* 1.2 (1972), p. 146-160 (cf. p. 10).

[55] Alan M Turing. « On computable numbers, with an application to the Entscheidungsproblem ». In : *Proceedings of the London mathematical society* 2.1 (1937), p. 230-265 (cf. p. 2).

[56] Andreas Weber et Helmut Seidl. « On the Degree of Ambiguity of Finite Automata ». In : *Theor. Comput. Sci.* 88.2 (1991), p. 325-349 (cf. p. 47, 48).

[57] D. Ziadi, J.-L. Ponty et J.-M. Champarnaud. « Passage d'une expression rationnelle a un automate fini non deterministe ». In : *Bulletin of the Belgian Mathematical Society - Simon Stevin* 4 (1997), p. 177-203 (cf. p. 35).

Index

Étude d'extensions des langages déterministes

Cette thèse a pour but d'étudier des propriétés structurelles d'automates étendant celle du déterminisme, et les langages pouvant être dénotés par une expression rationnelle dont l'automate des positions présente l'une de ces propriétés. Si Book *et al.* ont montré que tous les langages rationnels peuvent être reconnus par un automate des positions non-ambigu, Brüggemann-Klein et Wood ont montré que ceux pouvant l'être par un automate des positions déterministe forment une famille strictement incluse dans celle des rationnels. Nous nous intéressons aux extensions de cette famille, en cherchant à caractériser leurs langages, et à étudier leur hiérarchie interne et leur inclusion entre elles.

Mots-clés : Automates Finis, Langages Rationnels, Expressions Rationnelles, Automate des Positions.

Deterministic languages extensions

This thesis aims to study structural properties of automata extending determinism, and the languages that can be denoted by a regular expression of which the position automaton has one such property. If Book *et al.* showed that all regular languages can be recognized by an unambiguous position automaton, Brüggemann-Klein and Wood showed that only a proper subset of them can be recognized by a deterministic position automaton. We focus on extensions of this subfamily, by seeking to characterize their languages, and to study their internal hierarchy and how they relate to each other.

Keywords : Finite Automata, Regular Languages, Regular Expressions, Position Automaton.