Systèmes de Head et applications à la bio-informatique

Serghei Verlan

École doctorale IAEM Lorraine
Département de Formation Doctorale en Informatique

Université de Metz
UFR MIM

Systèmes de Head et applications à la bio-informatique

THÈSE

présentée et soutenue publiquement le 21 juin 2004

pour l'obtention du

Doctorat de l'Université de Metz

spécialité Informatique

par

Serghei Verlan

Composition du jury

Directeur	Maurice Margenstern	Professeur à l'Université de Metz
Rapporteurs	Tero Harju	Directeur de recherche à l'Université de Turku, Académie de Finlande
	Giancarlo Mauri	Professeur à l'Université de Milano-Bicocca, Italie
	Jean Néraud	Professeur à l'Université de Rouen
Examinateurs	Hoaï An Le Thi	Professeur à l'Université de Metz
	Hazel Everett	Professeur à l'Université de Nancy 2

Laboratoire d'Informatique Théorique et Appliquée

École doctorale IAEM Lorraine
Département de Formation Doctorale en Informatique

Université de Metz
UFR MIM

Systèmes de Head et applications à la bio-informatique

THÈSE

présentée et soutenue publiquement le 21 juin 2004

pour l'obtention du

Doctorat de l'Université de Metz

spécialité Informatique

par

Serghei Verlan

Composition du jury

Directeur	Maurice Margenstern	Professeur à l'Université de Metz
Rapporteurs	Tero Harju	Directeur de recherche à l'Université de Turku, Académie de Finlande
	Giancarlo Mauri	Professeur à l'Université de Milano-Bicocca, Italie
	Jean Néraud	Professeur à l'Université de Rouen
Examinateurs	Hoaï An Le Thi	Professeur à l'Université de Metz
	Hazel Everett	Professeur à l'Université de Nancy 2

Laboratoire d'Informatique Théorique et Appliquée

Remerciements

Tout d'abord, je voudrais remercier mon directeur de thèse, Maurice Margenstern, dont l'expérience scientifique m'a beaucoup guidé dans mon travail. Je le remercie également de sa patience avec moi et de son aide durant ces dernières années.

Je remercie énormément mon ancien professeur de Moldavie, Yurii Rogozhin, qui m'a initié au domaine des calculs moléculaires. Nos discussions lors de ses séjours à Metz ont beaucoup contribué d'une manière directe ou indirecte aux résultats dans ce travail et son soutien amical m'a été très précieux.

Je voudrais aussi remercier mes co-auteurs : Rudolf Freund, Franziska Freund, Marion Oswald, Gheorghe Păun et Rosalba Zizza pour des idées originales que je n'aurais peut-être jamais trouvé sans leur aide.

Je voudrais adresser un remerciement particulier à Gheorghe Păun qui m'a souvent inspiré de nombreuses idées.

Je remercie aussi toutes les personnes avec lesquelles on a passé des moments agréables aux conférences et qui m'ont suggéré souvent des idées intéressantes, en particulier Daniela Besozzi, Claudio Feretti, Pierluigi Frisco et Claudio Zandron.

Je remercie le projet OTAN PST.CLG.976912 et le projet IST-2001-32008 « MolCoNet » qui m'ont permis de participer aux conférences où j'ai pu présenter mes travaux et échanger d'expérience et d'idées avec d'autres participants. Je remercie aussi le Ministère de l'Éducation Nationale et de la Recherche de France pour le support financier de mon doctorat, ainsi que le Laboratoire d'Informatique Théorique et Appliquée de l'Université de Metz qui m'a accueilli et en particulier son secrétaire Damien Aignel qui m'a souvent aidé dans mes démarches administratives.

Je voudrais aussi remercier tous mes amis de Metz qui m'ont permis de me sentir ici comme chez moi.

Enfin, je remercie mes parents qui ont toujours cru en moi et qui m'ont soutenu pendant toutes ces années.

Table des matières

Introduction

Tout est nombre. En reprenant cette ancienne idée pythagoricienne et en la recombinant avec le fait qu'un nombre est le résultat d'un calcul, nous pouvons reformuler la phrase ci-dessus en disant que tout est calcul. Nous pouvons retrouver des exemples de calcul dans de nombreux domaines, mais l'exemple le plus fascinant est celui de la vie. Il n'est pas facile de donner une définition exacte à ce phénomène complexe, mais à l'heure actuelle on suppose que l'ingrédient le plus important de la vie est l'acide désoxyribonucléique, l'ADN, qui est parfois remplacé par l'acide ribonucléique, ou ARN. En effet, tous les êtres vivants que l'on connaît possèdent cette molécule et la nature utilise des opérations simples de type copier, coller, insérer, effacer etc. pour la manipuler. On voit que l'on a besoin seulement de petites briques, l'ADN, et de règles simples pour exprimer certains aspects structuraux des êtres vivants et peut-être même la diversité de la nature. On peut remarquer un parallèle avec la théorie de la calculabilité, où on utilise aussi des éléments très simples pour construire des fonctions qui calculent des choses complexes. Cette analogie nous conduit à la réflexion que la nature calcule, mais peut-être d'une manière différente que nous ne comprenons pas encore complètement. Du chaos à la matière inorganique, de la matière morte à la matière vivante, de l'inconscient à l'intelligence, peut-être toute l'évolution de l'univers est-elle une histoire de calculs qui deviennent de plus en plus compliqués. Tout cela paraît une spéculation métaphysique, mais, qui sait, peut-être notre conception actuelle des calculs est-elle liée à l'évolution de l'humanité, comme l'utilisation de la base dix pour compter est liée au fait que l'on a dix doigts. De même que les hommes ont commencé à utiliser d'autres bases pour compter, il est peut-être temps de trouver de nouvelles conceptions du calcul qui seront différentes de celles utilisées auparavant.

Un premier pas a été fait par L. Adleman en 1994 qui a montré pour la première fois comment il est possible d'utiliser ces nouvelles opérations et structures de données pour résoudre des problèmes concrets. Son expérience montre comment résoudre une instance du problème d'un chemin hamiltonien dans un graphe à l'aide de moyens biologiques uniquement. C'est incroyable, mais en utilisant des molécules d'ADN, des enzymes et d'autres substances chimiques et en manipulant des éprouvettes, il est possible de trouver une solution à un problème mathématique concret. L'expérience d'Adleman, publiée dans « Science », voir [1], a stimulé l'étude des nouveaux modèles de calcul fondés sur la biologie en provoquant une explosion d'articles liés à ce sujet. Divers aspects des mécanismes biologiques ont été étudiés d'un

point de vue de la calculabilité et les résultats sont très prometteurs, car les modèles obtenus permettent, bien sûr théoriquement, de dépasser la puissance de calcul des ordinateurs existants. La structure compacte des données, le parallélisme massif et la consommation modeste d'énergie, sont les atouts importants qui affirment la supériorité, au moins théorique, d'un ordinateur fondé sur des principes biologiques.

Nous remarquons que l'étude théorique des systèmes biologiques d'un point de vue de la calculabilité a commencé bien avant 1994. En effet, déjà en 1987, T. Head dans son article « Formal Language Theory and DNA : an Analysis of the Generative Capacity of Specific Recombinant Behaviors », [12], a montré des connexions entre la biologie moléculaire et la théorie des langages formels. Sa vision pionnière de la recombinaison moléculaire lui a permis d'apercevoir, longtemps avant l'expérience d'Adleman, l'importance et le potentiel énorme des calculs fondés sur des principes biologiques. Les systèmes qu'il a introduits et que nous appelons aujourd'hui des systèmes de Head, fondés sur l'opération de la recombinaison, représentent, même à présent, une branche importante du domaine. L'idée de Head était très simple : on remplace des molécules d'ADN par des mots et des enzymes par des règles de recombinaison. Puis, en mélangeant tout dans un tube à essai, on laisse le système évoluer. Mais, malgré ces travaux, nous pouvons dire que la naissance du domaine des calculs (bio)-moléculaires date de l'expérience d'Adleman qui a montré que les choses considérées auparavant comme fantastiques deviennent désormais réalisables.

Quels sont les buts dans ce domaine ? Bien sûr, le but le plus important est de construire un ordinateur fondé sur des principes biologiques. Mais, à présent, il est encore difficile de conjecturer quand nous pourrons travailler sur des ordinateurs de ce type. Un autre but qui est aussi très important, est de comprendre comment la nature calcule et de trouver peut-être de nouveaux paradigmes de calcul qui pourront être implantés, même sur des supports ordinaires. De même que l'étude du cerveau a permis de concevoir les réseaux neuronaux et l'étude de l'évolution a débouché sur la création des algorithmes génétiques, nous espérons apprendre de la nature de nouvelles méthodes de calcul que nous pourrions utiliser ensuite.

L'ADN n'est pas la seule source d'inspiration pour trouver de nouveaux paradigmes de calcul. Nous pouvons changer d'optique et nous placer au niveau cellulaire. En effet, la cellule vivante est une machine étonnante que la nature perfectionne de plus en plus. Tout est présent ici : la communication avec l'environnement, la reconnaissance des signaux, la recherche dans la base de données de l'ADN, la réaction, la production et la consommation d'énergie... En bref, la compréhension du comportement de la cellule est peut-être la clef qui nous permettra de comprendre les secrets de la vie. Mais, si on peut imaginer des calculs avec l'ADN, on peut aussi imaginer des calculs avec des cellules. Dans ce cas l'aspect structurel devient de plus en plus important, car la cellule peut être vue comme un ensemble de membranes imbriquées les unes dans les autres. Le premier modèle de calcul fondé sur la structure de la cellule et sur les processus cellulaires a été proposé en 1998 par Gh. Păun qui est considéré comme père de ce domaine : le calcul à membranes. Depuis, plus d'une centaine de variantes ont été proposées et leur nombre croît constamment. Vu le niveau technologique actuel, il n'y a pas encore d'implantation pratique *in vivo*

ou *in vitro*, mais comme nous l'avons dit auparavant, une des raisons pour lesquelles on étudie les modèles issus de la biologie est de trouver de nouvelles possibilités de calcul différentes de celles que nous avons utilisées jusqu'à présent.

D'une part, la théorie des langages formels est fondée sur l'opération de la réécriture. D'autre part, la nature utilise des opérations différentes du genre copier et coller, ainsi que des structures de données différentes. C'est pourquoi il est très important de pouvoir reconstruire les anciens paradigmes de calcul dans ce nouvel environnement. Même si ce n'est pas très important pour une réalisation pratique, cette reconstruction est cruciale pour des applications informatiques.

Nous nous arrêtons ici pour les spéculations métaphysiques et nous renvoyons à l'article [16] qui contient une introduction excellente dans le domaine, ainsi que beaucoup de réflexions philosophiques dont nous n'avons présenté ici qu'une petite partie. De même, les ouvrages [44] et [43] contiennent une présentation très détaillée des motivations et des attentes concernant le domaine des calculs moléculaires, du calcul à membranes et du calcul naturel en général.

La présente thèse essaie de répondre à certaines questions ci-dessus. Nous nous concentrons sur l'étude de certains modèles de calculs moléculaires et de calcul à membranes pour montrer leur équivalence avec des modèles classiques comme les grammaires formelles et les machines de Turing. Nous faisons même un pas en avant en montrant d'une manière directe, sans passer par les systèmes classiques, l'équivalence de certains modèles considérés. Finalement, nous avons conçu une méthode qui permet de simplifier la démonstration de ces équivalences, ainsi que la construction de systèmes spécifiques répondant à certains besoins. Nous remarquons que tous les modèles que nous considérons ont un trait commun : ils sont fondés sur l'opération de la recombinaison. Nous pouvons donc dire qu'on effectue une étude approfondie de cette opération qui, en effet, est très puissante par elle-même et, comme le montrent les nombreux exemples de ce travail, quelques petits ajouts suffisent pour qu'elle devienne aussi puissante que la réécriture.

Nous remarquons aussi que notre travail est purement théorique et que nous considérons l'aspect mathématique des calculs moléculaires et non pas l'aspect biologique.

Nous allons décrire maintenant le contenu de la thèse chapitre par chapitre.

Chapitre 1

Dans ce chapitre nous présentons brièvement certaines définitions de la théorie des langages formels pour fixer les notations que nous utilisons ensuite.

Chapitre 2

Nous introduisons d'une manière informelle les systèmes dont nous parlerons au cours de ce travail et nous donnons l'état de l'art.

Chapitre 3

Dans ce chapitre nous introduisons les notions centrales de notre thèse : l'opération de la recombinaison et les systèmes de Head. Nous considérons les deux définitions possibles de la recombinaison : la recombinaison simple et la recombinaison double et nous étudierons pour la première fois la relation entre les classes de langages fondés sur la recombinaison simple et sur la recombinaison double. Nous montrons qu'entre ces deux familles il existe une relation d'inclusion stricte. De même, dans ce chapitre nous présentons de nombreux exemples de systèmes de Head, ainsi que des langages rationnels qui ne peuvent pas être des langages de recombinaison.

Chapitre 4

Dans ce chapitre nous introduisons une extension des systèmes de Head : les systèmes de Head étendus et nous présentons la méthode « faire-tourner-et-simuler » qui est souvent utilisée pour démontrer une équivalence avec une grammaire formelle. De même, nous introduisons les systèmes distribués à changement de phase qui sont un modèle de calcul à la fois simple et puissant. Nous remarquons que la simplicité de la structure de ces systèmes permet de nombreuses utilisations, ce que nous avons souvent fait ensuite. Nous montrons qu'avec deux composants il est possible de produire tous les langages récursivement énumérables et qu'avec un seul composant il est possible de simuler le comportement d'une famille de systèmes de Post qui contient des systèmes universels. Nous remarquons que ce dernier système possède une description très compacte. Dans le même chapitre nous présentons des résultats obtenus à l'aide d'un logiciel de simulation des systèmes distribués à changement de phase à l'ordinateur, nommé `TVDHsim`. En effet, ce logiciel a permis de trouver des inexactitudes dans certains articles sur les systèmes distribués à changement de phase et nous présentons les systèmes correspondants tout en proposant des modifications pour leur redonner un comportement correct dans les cas où c'était possible. Nous ajoutons que ce même logiciel nous a permis d'acquérir une grande expérience de construction des systèmes fondés sur la recombinaison en soulignant des pièges cachés dans lesquelles plusieurs auteurs sont tombés auparavant. Nous remarquons enfin que tous les systèmes distribués à changement de phase présents dans ce travail ont été contrôlés à l'aide de ce logiciel.

Chapitre 5

Le chapitre 5 introduit une extension des systèmes distribués à changement de phase : les systèmes distribués à changement de phase étendus qui possèdent un comportement plus complexe. Nous montrons que trois composants sont suffisants pour produire tous les langages récursivement énumérables.

Chapitre 6

Le chapitre 6 contient une analyse des techniques de démonstration utilisées au cours des deux chapitres précédents. Cette analyse débouche sur une nouvelle mé-

thode, la méthode des molécules assistantes, qui permet, d'une manière inattendue, de diminuer la complexité des systèmes correspondants en montrant qu'ils possèdent un contrôle additionnel que nous avons mis en évidence. En utilisant ce contrôle intrinsèque nous pouvons diminuer facilement le nombre de composants nécessaires pour obtenir la puissance d'expression d'une machine de Turing. Ainsi, nous arrivons aux limites de ces systèmes en montrant pour eux la frontière entre la décidabilité et l'indécidabilité. À la fin de ce chapitre nous proposons une généralisation de la méthode des molécules assistantes que nous allons utiliser ensuite aux chapitres 7 et 8.

Chapitre 7

Dans ce chapitre nous considérons une autre extension des systèmes de Head : les systèmes de tubes à essai. Nous montrons que le problème ouvert de la puissance d'expression des systèmes de tubes à essai avec deux tubes a stimulé l'apparition de nombreuses variantes de ces systèmes. Nous introduisons ici une nouvelle variante : les systèmes de tubes à essai avec filtres alternants et nous montrons que dans ce cas deux tubes suffisent pour produire tous les langages récursivement énumérables. De plus, nous pouvons placer les règles du système de façon à ne pas avoir de règle dans le deuxième tube qui est utilisé uniquement comme une poubelle. En outre, ces systèmes permettent de montrer que les systèmes de Head sont déjà assez puissants et il suffit d'y apporter de petites modifications pour passer de la rationalité à l'universalité. La méthode des molécules assistantes joue un rôle important dans ce chapitre et la plupart des démonstrations l'utilisent pleinement.

Chapitre 8

Ce chapitre comporte une autre variante des systèmes de tubes à essai : les systèmes de tubes à essai modifiés. Un petit détail différencie ces systèmes, mais il est suffisant pour atteindre la puissance d'expression d'une machine de Turing avec deux tubes uniquement. Nous remarquons que ce résultat est très difficile à obtenir sans utiliser la méthode des molécules assistantes, car le système correspondant possède un comportement extrêmement complexe. De même, le concept de poubelle mobile, quand les molécules indésirables évitent les règles qui peuvent leur être appliquées, nécessite une bonne synchronisation entre différentes parties du calcul.

Chapitre 9

Dans ce chapitre nous nous concentrons sur les systèmes à membranes, en particulier sur les variantes qui utilisent l'opération de la recombinaison. Nous montrons que la définition originale des systèmes à membranes avec recombinaison possède des lacunes et nous proposons plusieurs solutions. Nous montrons aussi que la puissance d'expression de ces systèmes ayant une seule membrane dépend de la solution choisie. Dans le même chapitre, nous considérons d'autres variantes des systèmes à

membranes fondés sur la recombinaison : les systèmes à membranes avec recombinaison non-étendus et les systèmes à membranes avec recombinaison et communication immédiate et nous montrons comment il est possible de diminuer leur paramètre de complexité principal : le nombre des membranes. Nous présentons une solution définitive, car nous montrons une frontière entre la décidabilité et l'indécidabilité pour ces systèmes. Nous montrons aussi que la méthode des molécules assistantes est applicable dans ce nouvel environnement.

Finalement, nous présentons quelques conclusions et quelques problèmes ouverts.

Chapitre 1

Prérequis

Dans ce chapitre nous fixons les notations que nous allons utiliser par la suite. Pour plus de détails concernant la théorie des langages formels voir [15] et [47].

1.1 Mots et langages

Nous notons $\mathbb{N}$ l'ensemble des entiers naturels $\{0, 1, 2, \dots\}$ et $\mathbb{N}^+$ l'ensemble des entiers strictement positifs $\{1, 2, \dots\}$. Nous notons aussi $\emptyset$ l'ensemble vide et $\mathcal{P}(X)$ l'ensemble de tous les sous-ensembles de X. Le nombre d'éléments d'un ensemble X est noté $Card(X)$.

Un *alphabet* est un ensemble fini et non-vide de symboles. Un *mot* sur un alphabet V est une concaténation de symboles, dits aussi *lettres*, de V. La concaténation vide est appelé le *mot vide* et elle est notée ε. L'ensemble de tous les mots sur V est noté V^*. L'ensemble de tous les mots sur un alphabet V, excepté le mot vide, est noté V^+. Tout sous-ensemble de V^* est appelé *langage* sur l'alphabet V.

Si $x = x_1x_2$ avec $x_1, x_2 \in V^*$, on dit que le mot x_1 est un *préfixe* de x et que le mot x_2 est un *suffixe* de x. Si $x = x_1x_2x_3$ avec $x_1, x_2, x_3 \in V^*$, x_2 est appelé facteur de x. L'ensemble des préfixes, suffixes et facteurs d'un mot x est noté $Pref(x)$, $Suf(x)$ et $Fact(x)$, respectivement.

La *longueur* du mot $x \in V^*$ est le nombre de symboles qui apparaissent dans x et elle est notée $|x|$. Le nombre des occurrences d'un symbole $a \in V$ dans $x \in V^*$ est noté $|x|_a$. Si $x \in V^*$ et $U \subseteq V$, on note $|x|_U$ le nombre d'occurrences des symboles de U dans x.

On note les opérations booléennes sur les langages d'une manière habituelle : l'intersection est notée $\cap$, l'union $\cup$ ou $+$ et la différence $\setminus$.

La *concaténation* de deux langages L_1 et L_2 est définie par :

$$L_1L_2 = \{xy : x \in L_1, y \in L_2\}.$$

On définit l'itération de la concaténation de la manière suivante :

$$L^0 = \{\varepsilon\},$$
$$L_{i+1} = LL^i, \quad i \geq 0,$$
$$L^* = \bigcup_{i=0}^{\infty} L^i \quad \text{(la clôture de Kleene).}$$

On dit qu'une famille de langages $\mathcal{F}$ est fermée par rapport à une opération n-aire g si le résultat de l'application de g aux langages $L_1, \ldots, L_n$ appartenant à $\mathcal{F}$ appartient aussi à $\mathcal{F}$: $g(L_1, \ldots, L_n) \in \mathcal{F}$.

1.2 Grammaires formelles

Une *grammaire* formelle est un quadruplet $G = (N, T, S, P)$, où N et T sont deux alphabets disjoints, $S \in N$, l'axiome, et où P est un ensemble fini des règles de la forme $u \to v$, $u, v \in (N \cup T)^*$ avec $|u|_N > 0$. L'alphabet N, respectivement T, est appelé l'alphabet non-terminal, respectivement terminal, de G. Les règles de P sont appelées aussi des *productions.*

Pour $x, y \in (N \cup T)^*$ on écrit :
$x \underset{G}{\Rightarrow} y$ si
$x = x_1 u x_2$, $y = x_1 v x_2$ avec $x_1, x_2 \in (N \cup T)^*$ et s'il existe une production $u \to v \in P$.

Dans ce cas on dit que x produit directement y. Nous pouvons omettre G si le contexte le permet. Nous notons $\overset{*}{\Rightarrow}$ la clôture réflexive et transitive de $\Rightarrow$. Nous écrivons $x \overset{k}{\Rightarrow} y$ s'il existe des mots $w_0, \ldots, w_k$ tels que $w_i \Rightarrow w_{i+1}, i \geq 0$ et $w_0 = x$ et $w_k = y$. Nous écrivons $x \underset{P'}{\overset{*}{\Rightarrow}} y$, où P' est un sous-ensemble de P, si pour obtenir y à partir de x nous utilisons des productions faisant partie de P' uniquement.

Chaque mot $w \in (N \cup T)^*$ dérivé à partir de l'axiome de la grammaire G : $S \underset{G}{\overset{*}{\Rightarrow}} w$ est appelé *forme syntaxique* de G.

Le langage produit par G, L(G), est défini par :

$$L(G) = \{w \in T^* : S \overset{*}{\Rightarrow} w\}.$$

Selon la forme de leur règles les grammaires formelles peuvent être de différents types.

- *Grammaires arbitraires (de type 0) :*
  Les productions sont de la forme $u \to v$, où $u, v \in (N \cup T)^*$ et $|u|_N > 0$.
- *Grammaires contextuelles (de type 1) :*
  Les productions sont de la forme $u_1 A u_2 \to u_1 x u_2$, où $u_1, u_2 \in (N \cup T)^*$, $A \in N$ et $x \in (N \cup T)^+$.
- *Grammaires algébriques (de type 2) :*
  Les productions sont de la forme $A \to x$, où $A \in N$ et $x \in (N \cup T)^*$.
- *Grammaires rationnelles (de type 3) :*
  Les productions sont soit de la forme $A \to aB$ et $A \to a$, soit de la forme $A \to Ba$ et $A \to a$, où $A, B \in N$ et $a \in T$.

Nous notons respectivement RE, CS, CF et REG la famille des langages produits par, respectivement, les grammaires arbitraires, contextuelles, algébriques et rationnelles. Les familles RE, CS, CF et REG sont respectivement appelées la famille des langages récursivement énumérables, contextuels, algébriques et rationnels, respectivement. Ces familles forment la hiérarchie de Chomsky. Nous notons aussi FIN la famille des langages finis. Les inclusions strictes suivantes ont lieu.

$$FIN \subset REG \subset CF \subset CS \subset RE.$$

Les propriétés de fermeture des familles ci-dessus par rapport aux opérations sur les langages sont montrées au tableau 1.1.

TAB. 1.1 – Les propriétés de fermeture des familles de la hiérarchie de Chomsky.

Opération	**RE**	**CS**	**CF**	**REG**
Union	Oui	Oui	Oui	Oui
Intersection	Oui	Oui	Non	Oui
Concaténation	Oui	Oui	Oui	Oui
Clôture de Kleene	Oui	Oui	Oui	Oui
Intersection avec REG	Oui	Oui	Oui	Oui

Un *système de Post* simplifié de degré $m > 0$, voir [4] et [32], est un triplet $T = (m, V, P)$, où $V = \{a_1, \ldots, a_{n+1}\}$ est un alphabet et où P est un ensemble de productions de la forme $a_i \to P_i, 1 \leq i \leq n$, $P_i \in V^*$. Le symbole a_{n+1} est appelé le symbole d'arrêt. Une configuration du système T est un mot w. Nous passons d'une configuration $w = a_{i_1} \ldots a_{i_m} w'$ à la configuration suivante z en éliminant les m premiers symboles de w et en ajoutant P_{i_1} à la fin du mot : $w \Rightarrow z$, si $z = w' P_{i_1}$.

Le calcul de T sur un mot $x \in V^*$ est une suite de configurations $x \Rightarrow \ldots \Rightarrow y$, où soit $y = a_{n+1} a_{i_1} \ldots a_{i_{m-1}} y'$, soit $y' = y$ et $|y'| < m$. Dans ce cas on dit que T s'arrête sur x et que y' est le résultat du calcul de T sur x. On dit que T reconnaît un langage L si pour tout $x \in L$, T s'arrête sur x et T ne s'arrête que sur les mots de L.

Nous ajoutons que les systèmes de Post simplifiés de degré 2 sont capables de reconnaître la famille des langages récursivement énumérables, voir [4] et [32].

Nous rappelons une notion importante d'une *constante* pour un langage rationnel L qui a été définie par Schutzenberger dans [48]. Nous présentons une définition équivalente, mais qui est plus simple, donnée dans [12, 13].

Définition 1.2.1. [12, 13] Un mot $w \in V^*$ est une *constante* pour un langage L si toutes les fois que les mots xwy et uwv font partie de L, les mots xwv et uwy font aussi partie de L.

Nous allons utiliser souvent la remarque suivante.

Remarque 1.2.1. Soit $L \subseteq V^*$ un langage rationnel. Il est clair que si $c \notin V$, le mot c est une constante pour les langages Lc, cL et LcL. Il est facile de voir aussi que si $c, d \notin V$, les mots c et d sont des constantes pour le langage $cL + Ld$. Finalement, l'ensemble $\{ca : a \in V\}$ est un ensemble fini de constantes pour le langage cLc.

1.3 Automates finis et machines de Turing

Un automate fini est un quintuplet $M = (Q, V, q_0, F, \delta)$, où Q est un ensemble fini d'*états*, où V est un ensemble fini de symboles, où $q_0 \in Q$ est l'état initial, où $F \subseteq Q$ est l'ensemble des états finaux et où $\delta : Q \times V \to \mathcal{P}(Q)$ est la fonction de transition de l'automate. Nous définissons l'opération de transition $\vdash$ d'une manière ordinaire : $(s, ax) \vdash (s', x)$ si $s' \in \delta(s, a)$, où $s, s' \in Q$, $a \in V$ et $x \in V^*$. Nous notons par $\vdash^*$ la fermeture réflexive et transitive de $\vdash$.

Le mot x est dit accepté par M si $(q_0, x) \vdash^* (q, \varepsilon)$ et $q \in F$. Le langage accepté par M est :

$$L(M) = \{x \in V^* : (q_0, x) \vdash^* (q, \varepsilon), q \in F\}.$$

Nous utiliserons aussi une notation graphique pour définir un automate fini. Dans ce cas, un automate fini sera représenté par un graphe orienté et étiqueté dont les noeuds représentent les états de l'automate et dont les arêtes sont définies par la fonction de transition de l'automate. Plus précisément, il existe une arête marquée par a entre les noeuds q_i et q_j du graphe si $q_j \in \delta(q_i, a)$. Les états finals sont entourés d'un double cercle.

Une machine de Turing est un 6-uplet $M = (Q, T, a_0, q_0, F, \delta)$, où Q est un ensemble fini d'états, T est l'alphabet du ruban, où $a_0 \in T$ est le symbole vide, où $q_0 \in Q$ est l'état initial, où $F \subseteq Q$ est l'ensemble d'états finaux et où δ est une fonction de transition. Chaque règle de δ, appelée aussi instruction, est de la forme $q_i a_k D a_l q_j$, où $q_i, q_j \in Q$, $a_k, a_l \in T$ et où $D \in \{L, S, R\}$. La sémantique de la règle $q_i a_k D a_l q_j$ est la suivante : si dans l'état q_i la tête de la machine voit le symbole a_k sur le ruban, elle change son état en q_j, remplace a_k par a_l et se déplace d'un pas à gauche si $D = L$, d'un pas à droite si $D = R$ ou reste sur place si $D = S$. Nous remarquons que pour chaque machine de Turing qui possède des instructions stationnaires, c'est-à-dire sans déplacement, il est possible de construire une machine équivalente qui n'aura aucune instruction stationnaire.

Une *configuration* de la machine de Turing M est le mot suivant $w_1 q_i a_k w_2$, où $w_1 a_k w_2$ est la partie du ruban qui contient des symboles non-vides, q_i est l'état de la machine et a_k est la case qui est examinée par la tête de la machine.

Un *calcul* de la machine de Turing M sur un mot $x \in T^+$ est une suite de configurations $q_0 x \Rightarrow \ldots w_1 q_f w_2$, où $q_f \in F$. La configuration initiale peut être de la forme $u_1 q_0 u_2$, où $x = u_1 u_2$, mais nous pouvons supposer sans perte de généralité qu'elle est $q_0 x$. Dans ce cas nous disons que $w_1 w_2$ est le résultat du calcul de M sur le mot x.

Chapitre 2

L'état de l'art

2.1 L'opération de recombinaison

L'expérience d'Adleman, voir [1], a lancé l'étude des différentes opération biologiques d'un point de vue de la calculabilité. Les opérations qu'il a utilisé : synthèse, amplification, séparation, extraction et détection, ainsi que des opérations similaires ont été reprises par des divers auteurs qui ont montré comment il est possible de résoudre plusieurs problèmes mathématiques à l'aide de ces opérations. Par exemple, des solutions polynomiales pour SAT, DES et d'autres problèmes ont été proposées, voir [19]. Une caractéristique importante de toutes ces opérations est qu'elles correspondent à de vraies opérations biologiques, par conséquent, il est possible de les implanter réellement. D'autre part, leur formalisation est difficile, car beaucoup de paramètres quantitatifs d'origine physique doivent être pris en compte. C'est pourquoi certains auteurs ont choisi une autre approche en formalisant des opérations biologiques, mais en idéalisant les contraintes d'origine physique. Les opérations obtenus, comme insertion, effacement, recombinaison etc. présentent de nombreux avantages théoriques, mais leur réalisation en pratique reste difficile. Nous remarquons que les deux approches sont intéressantes, mais nous nous concentrons surtout sur la deuxième. Plus précisément, nous étudions l'opération de la recombinaison et des divers systèmes fondés sur cette opération.

Nous précisons que l'étude théorique de la recombinaison a commencé bien avant l'expérience d'Adleman. En 1987 déjà, dans son article [12], T. Head a trouvé une connexion inattendue entre la biologie moléculaire et la théorie des langages formels. En formalisant le processus de l'action des enzymes de restriction et de ligation, il a obtenu une opération de recombinaison définie sur des chaînes de caractères. Ainsi les systèmes de recombinaison, ou systèmes de Head, ont été introduits comme un nouveau moyen de production de langages.

Nous ne présenterons pas ici toutes les réflexions nécessaires pour passer de l'événement biologique jusqu'à l'opération formelle, tous les détails pouvant être retrouvés dans [44, 12] ; nous commencerons directement par la description formelle. Une règle de recombinaison sur un alphabet A est le quadruplet suivant $(u_1, u_2; u_3, u_4), u_i \in A^*$ que nous noterons aussi de la façon suivante : $u_1\#u_2\$u_3\#u_4$,

où $\{\#, \$\} \notin A$. Nous appliquons cette règle à une paire de mots, dits aussi molécules, $x = x_1u_1u_2x_2$ et $y = y_1u_3u_4y_2$ et nous obtenons deux mots : $w = x_1u_1u_4y_2$ et $z = y_1u_3u_2x_2$. On cherche donc dans les mots initiaux les sous-chaînes indiquées dans la règle de recombinaison, qui sont appelés des sites de la recombinaison. Puis, les chaînes initiales sont coupées à l'endroit précisé par la règle et leurs extrémités sont échangées et collées. Plus exactement, l'opération de recombinaison échange les extrémités de deux mots, l'endroit où on effectue l'échange étant indiqué par la règle. Nous notons ce fait par $(x, y) \vdash_r (w, z)$. Dans ce cas nous parlons aussi de la recombinaison double. Si nous considérons seulement w comme résultat, nous parlons de la recombinaison simple. La recombinaison double est une opération plus faible que la recombinaison simple et peut être facilement simulée par celle-ci. C'est pourquoi la recombinaison double est généralement utilisée dans la littérature, car tous les résultats obtenus pour la recombinaison double peuvent être facilement traduits en termes de recombinaison simple.

La définition de la recombinaison présentée plus haut est celle proposée par Gh. Păun dans [37]. Il existe encore deux définitions de la recombinaison, une proposée par T. Head dans [12] et l'autre proposée par D. Pixton dans [33]. Ces définitions comportent des propriétés similaires ; une comparaison de la puissance d'expression de ces trois définitions peut être trouvée dans [61]. La définition la plus répandue et généralement acceptée est celle de Păun et nous nous concentrerons dans ce qui suit sur celle-ci.

Nous observons qu'une molécule peut contenir plusieurs sites identiques qui contiennent les sous-chaînes de la règle de recombinaison. Un de ces sites est alors choisi d'une manière non-déterministe pour effectuer la recombinaison. Puis nous supposons que le nombre de molécules x et y qui participent à la recombinaison n'est pas borné, x et y peuvent donc être utilisés à nouveau pour une recombinaison, c'est à dire qu'ils ne sont pas consommés par la recombinaison. Cette considération est d'origine physique, car dans quelques grammes d'une solution il y a un nombre de molécules de l'ordre 10^{22}.

Soit V un alphabet et R un ensemble de règles de recombinaison. On dit alors que $\sigma = (V, R)$ est un schéma de recombinaison. Maintenant nous pouvons définir l'opération de recombinaison $\sigma(L)$ sur un langage L qui est le résultat de toutes les recombinaisons possibles des mots de L. Les propriétés de la recombinaison sur les langages sont discutées en détail au chapitre 7.2 de [44].

2.2 Les systèmes de Head

Maintenant, nous allons montrer comment il est possible d'utiliser l'opération de recombinaison comme moyen de production de langages.

Nous pouvons définir une itération de $\sigma(L)$ de la manière suivante :
$\sigma^0(L) = L$,
$\sigma^{i+1}(L) = \sigma^i(L) \cup \sigma(\sigma^i(L)),\ i \geq 0$,
$\sigma^*(L) = \cup_{i \geq 0} \sigma^i(L)$.

Un système de Head est le couple $H = (\sigma, A)$ où σ est un schéma de recombinaison et $A \subseteq V^*$ un ensemble de mots appelés axiomes. Le langage produit par un tel système est $L(H) = \sigma^*(A)$. Le langage d'un système de Head est donc le résultat d'une application itérative de l'opération de recombinaison sur l'ensemble des axiomes. Intuitivement le travail d'un système de recombinaison correspond à l'expérience biologique suivante : on mélange dans une éprouvette des molécules d'ADN, qui correspondent aux axiomes, et des enzymes, qui correspondent aux règles de recombinaison, et on attend jusqu'à ce qu'il n'y ait plus de réaction. D'autre part, le procédé décrit ci-dessus permet de produire des langages à partir d'un ensemble fini de mots initiaux et à partir d'un ensemble fini de règles de recombinaison.

Ces systèmes ont été considérés pour la première fois par T. Head qui a réussi à montrer des propriétés intéressantes. Plus tard, K. Culik II et T. Harju dans [6] ont montré que la puissance d'expression de ces systèmes est bornée par la famille des langages rationnels. Puis, des systèmes de Head dont l'ensemble d'axiomes ou celui des règles n'est pas fini ont été considérés. Si on note par $H(\mathcal{F}_1, \mathcal{F}_2)$ la famille de langages produits par les systèmes de Head dont les axiomes font partie de la famille $\mathcal{F}_1$ et dont les règles de recombinaison font partie de la famille $\mathcal{F}_2$, on connaît les résultats suivants présentés dans le tableau 2.1 ci-après, voir aussi [44].

TAB. 2.1 – La puissance d'expression des familles $H(\mathcal{F}_1, \mathcal{F}_2)$

	FIN	REG	LIN	CF	CS	RE
FIN	FIN, REG	FIN, RE	FIN, RE	FIN, RE	FIN, RE	FIN, RE
REG	REG	REG, RE	REG, RE	REG, RE	REG, RE	REG, RE
LIN	LIN, CF	LIN, RE	LIN, RE	LIN, RE	LIN, RE	LIN, RE
CF	CF	CF, RE	CF, RE	CF, RE	CF, RE	CF, RE
CS	CS, RE	CS, RE	CS, RE	CS, RE	CS, RE	CS, RE
RE	RE	RE	RE	RE	RE	RE

Dans le tableau, à l'intersection de la ligne $\mathcal{F}_1$ et la colonne $\mathcal{F}_2$, est placée la famille $H(\mathcal{F}_1, \mathcal{F}_2)$. S'il y a deux éléments $\mathcal{F}_3$ et $\mathcal{F}_4$ dans la case correspondante, on a l'inclusion stricte $\mathcal{F}_3 \subset H(\mathcal{F}_1, \mathcal{F}_2) \subset \mathcal{F}_4$.

Nous soulignons trois résultats de ce tableau :

$H(FIN, FIN) \subset REG$,

$H(REG, FIN) = REG$,

$H(FIN, REG) \subset RE$.

Ces inclusions montrent les limites de la recombinaison itérative. Même si nous utilisons un ensemble rationnel d'axiomes nous ne pouvons pas franchir le seuil de la rationalité dans le cas où un ensemble fini de règles est utilisé. La situation change si un ensemble rationnel de règles est permis. Mais cette approche satisfaisant les demandes d'un théoricien semble être loin du modèle biologique qui suppose que ces deux paramètres soient finis. De plus, en affirmant que les systèmes de Head sont un moyen de production de langages, comme les grammaires formelles ou les machines de Turing, nous supposons qu'un langage infini est produit à partir de paramètres finis. Nous parlerons plus loin de certaines des solutions qui auront un nombre fini

de composants tout en gardant une grande puissance d'expression, mais celles-ci nécessitent un contrôle additionnel.

La première inclusion ci-dessus a été montrée pour la première fois dans [6] en utilisant des propriétés algébriques des langages de recombinaison. Puis D. Pixton dans [33] a simplifié la preuve en utilisant des automates finis et elle peut être retrouvée avec tous les détails dans [14]. Une nouvelle simplification de la preuve pour le cas de la définition de Păun a été donnée dans [44]. Récemment, V. Manca dans [20] a proposé une preuve fondée sur la décomposition de l'opération de la recombinaison en trois opérations : couper, coller et effacer.

La même inclusion pose le problème de la caractérisation de la famille des langages produits par un système de recombinaison ayant des composants finis. Ce problème n'est pas résolu jusqu'à présent mais il existe des solutions partielles. Par exemple, T. Head dans [13] a caractérisé la famille des langages produits par des systèmes de recombinaison ayant des règles de la forme $u\#\varepsilon\$v\#\varepsilon$ ou $\varepsilon\#u\$\varepsilon\#v$. Ces systèmes présentent une connexion intéressante avec le concept de constante pour un langage rationnel introduit par Schutzenberger dans [48]. Plus précisément, les mots qui apparaissent comme sites des règles de recombinaison sont des constantes pour le langage produit. Dans le même article [13], on trouve des connexions intéressantes avec la théorie des automates finis.

Des caractérisations de classes de langages de recombinaison sont présentées dans [12, 11]. S.M. Kim dans [17] a donné une réponse partielle pour le cas de la variante de Head de la définition de la recombinaison. Une condition suffisante pour qu'un langage rationnel soit un langage de recombinaison est donnée dans la thèse de R. Zizza [61] et à présent un article sur ce sujet est en cours de publication. Dans la même thèse, un aperçu des travaux dans cette direction est donné.

Un autre problème qui n'a pas été étudié est celui de la comparaison entre les systèmes de Head fondés sur la recombinaison simple et ceux fondés sur la recombinaison double. Une première remarque que la recombinaison double peut être imitée par la recombinaison simple a été faite dans [44]. Mais on ne savait pas si cette inclusion est stricte. Nous avons étudié ce problème et nous avons prouvé dans [56] en collaboration avec R. Zizza que l'inclusion est stricte, la preuve pouvant être trouvée au chapitre 3 de cette thèse.

2.3 Extensions des systèmes de Head

L'étude des systèmes de Head a montré vite leurs limites, c'est pourquoi diverses extensions en ont été proposées. Un premier pas a été d'introduire un alphabet terminal. Un système de Head étendu est un système de Head H muni d'un alphabet terminal T. Le langage produit par un tel système consiste en tous les mots produits par H qui font partie de l'alphabet terminal. Dans le cadre de ce modèle, il est possible de produire tous les langages rationnels en partant d'un nombre fini d'ingrédients.

Une autre idée proposée ensuite est de considérer deux ensembles finis de sous-chaînes associées à chaque règle. L'application de la règle est contrôlée par ces

ensembles de la manière suivante : l'application est permise sur deux mots si et seulement si tous les éléments du premier ensemble apparaissent, respectivement n'apparaissent pas, dans le premier mot et tous les éléments du deuxième ensemble apparaissent, respectivement n'apparaissent pas, dans le second mot. Pour le reste, le fonctionnement ressemble à celui des systèmes de Head étendus. Les systèmes obtenus portent le nom de systèmes de recombinaison avec contexte permissif, respectivement restrictif. Il est possible d'atteindre la puissance d'expression d'une machine de Turing en utilisant ces modèles, aussi bien permissifs que restrictifs.

En même temps que les systèmes de Head étendus un autre modèle qui leur ressemble a été proposé. En effet, ce modèle utilise des multi-ensembles de mots à la place d'ensembles. À nouveau il est possible de simuler une grammaire quelconque et de produire donc tous les langages récursivement énumérables.

Il existe d'autres modèles qui sont des modifications des systèmes de Head et qui permettent d'obtenir la puissance d'expression d'une machine de Turing. Un aperçu de ces modèles et de leurs propriétés peut être trouvé au chapitre 8 de [44].

2.4 Systèmes de tubes à essai

Comme nous l'avons dit un peu plus tôt, les systèmes de Head ne possèdent pas par eux-mêmes une grande puissance de calcul. Afin de simuler de plus près les phénomènes biomoléculaires et d'obtenir une puissance de calcul suffisante, un nouvel ingrédient a été apporté à ces systèmes qui consiste à introduire plus ou moins de distribution dans les calculs. La plupart des modèles s'inspirent à la fois des systèmes de Head et des grammaires distribuées, voir [47]. Nous nous concentrons sur certains d'entre eux, un aperçu des modèles existants est donné dans [44].

Un des premiers modèles s'inspirant des grammaires distribuées est le modèle de tubes à essai introduit par E. Csuhaj-Varjú, L. Kari et G. Păun dans [5]. Ce modèle introduit des tubes à essai qui sont composés d'un ensemble d'axiomes, d'un ensemble de règles de recombinaison et d'un ensemble de lettres appelé filtre. Le calcul d'un tel système qui comporte un nombre fini de tubes, consiste en deux étapes répétées itérativement. Pendant la première étape, celle du calcul, chaque tube évolue comme un système de Head ordinaire. Pendant la deuxième étape, celle de la communication, les molécules présentes dans chacun des tubes sont envoyées vers tous les tubes, et les molécules qui franchissent le filtre d'un tube, c'est-à-dire qui sont composées uniquement des lettres qui forment l'ensemble du filtre, restent dans le tube correspondant et forment le langage initial pour l'étape suivante du calcul. Si une molécule peut franchir plusieurs filtres, chacun des tubes correspondants reçoit une copie de cette molécule. Les molécules ne pouvant franchir aucun filtre restent dans le tube dans lequel elles ont été produites. Le langage produit par un système de tubes à essai consiste en tous les mots produits par le système qui sont dans le premier tube et qui font partie d'un alphabet terminal.

Dans l'article introduisant ces systèmes, leur universalité a été montrée sans indiquer de limite sur le nombre de tubes. Puis, dans [59], les auteurs ont montré que 10 tubes suffisent pour pour produire tous les langages récursivement énumérables.

Un peu plus tard les mêmes auteurs ont montré que 9 tubes suffisent [60]. Le nombre de tubes nécessaires pour obtenir la puissance d'expression d'une machine de Turing a été réduit jusqu'à 6, voir [39], et finalement établi à 3, voir [34]. Nous remarquons qu'en ayant un seul tube, nous obtenons les systèmes de Head étendus et nous ne pouvons donc produire que des langages rationnels. Le problème de la puissance des systèmes de tubes à essai avec deux tubes n'est pas résolu jusqu'à présent, mais Gh. Păun suppose dans [44] que les langages produits par de tels systèmes appartiennent à la famille des langages algébriques.

Ce dernier problème a stimulé l'apparition de variantes des systèmes de tubes à essai. Dans la plupart des cas, les changements interviennent seulement au niveau du filtrage. Par exemple, dans la variante proposée par R. Freund et F. Freund dans [8], les filtres sont des unions finies d'ensembles et une molécule franchit un filtre si et seulement si elle est composée des éléments des ensembles et à condition que tous les éléments du même ensemble soient présents.

Le modèle obtenu est puissant et, dans ce cas, deux tubes suffisent pour produire tous les langages récursivement énumérables.

Une autre variante a été proposé par P. Frisco et C. Zandron dans [10]. Cette variante possède un filtre spécial à deux lettres qui contient des lettres ou des couples de lettres. Une molécule franchit le filtre si et seulement si elle est composé des lettres du filtre à condition que les deux lettres d'un couple apparaissent dans la molécule. À nouveau deux tubes permettent d'avoir la même puissance d'expression que les machines de Turing.

Nous avons introduit dans [55] encore une variante de systèmes de tubes à essai : les systèmes de tubes à essai avec filtres alternants. Dans cette variante, les filtres classiques sont remplacés par des n-uplets de filtres, chaque filtre d'un n-uplet restant un simple ensemble de lettres. A chaque itération, le filtre suivant du n-uplet est considéré comme filtre actif, c'est-à-dire à utiliser. Quand le dernier filtre du n-uplet est utilisé le premier filtre devient ensuite actif.

A nouveau deux tubes à essai avec deux filtres suffisent pour produire tous les langages récursivement énumérables. Il est possible de faire en sorte que les deux filtres du premier couple coïncident ou que les deux filtres de chaque couple diffèrent seulement par une lettre. Nous avons aussi obtenu un résultat inattendu : il est possible de produire tous les langages récursivement énumérables avec deux tubes où le deuxième tube ne contient pas de règles et qui est utilisé seulement comme une poubelle, l'universalité est donc obtenue avec « un tube et demi » ! Un autre apport de ce résultat est qu'il montre que les systèmes de Head sont déjà assez puissants et qu'il suffit d'y apporter de petites modifications pour passer de la rationalité à l'universalité. Tous ces résultats se trouvent au chapitre 7 de cette thèse.

Nous avons proposé encore une variante des systèmes de tubes à essai. Cette variante, les systèmes de tubes à essai modifiés, diffère de la définition originale par un « simple » détail : si une molécule peut franchir le filtre d'un tube différent de celui dans lequel elle a été produite, alors elle ne peut plus rester dans le tube initial même si elle peut franchir son filtre. Cette différence, en apparence pas très importante, permet de contrôler l'élimination des mots du tube et, par conséquent,

permet aux systèmes de tubes à essai modifiés de simuler les grammaires arbitraires. Les résultats concernant ces systèmes se trouvent au chapitre 8 de ce travail.

Nous remarquons que la démonstration de ces résultats sur les variantes des systèmes de tubes à essai utilise la méthode des molécules assistantes qui est décrite dans notre chapitre 6.

2.5 Systèmes distribués à changement de phase

Peu après l'introduction des systèmes de tubes à essai une autre possibilité de distribution des calculs a été considérée par Gh. Păun dans [39]. Il est parti de l'observation biologique suivante : à chaque moment, il y a un groupe d'enzymes actifs qui agissent en fonction des conditions de l'environnement. Si l'environnement change (température, acidité, ou autre paramètre), le jeu d'enzymes actifs change aussi. Dans le modèle proposé, l'ensemble des règles de recombinaison change périodiquement. Plus exactement, le modèle comporte un ensemble de mots, les axiomes, et un nombre fini d'ensembles de règles de recombinaison, les composants. À chaque étape du calcul, les mots présents se recombinent une fois selon l'ensemble courant de règles et le résultat de cette recombinaison forme l'ensemble des mots pour l'itération suivante. Nous remarquons que cette condition d'élimination est très forte et qu'elle permet d'obtenir une grande puissance de calcul. Les systèmes obtenus portent le nom de systèmes distribués à changement de phase.

Initialement Gh. Păun a montré que 7 composants sont suffisants pour obtenir tous les langages récursivement énumérables, voir [40, 44]. Son résultat a été amélioré par M. Margenstern et Yu. Rogozhin qui ont montré d'abord que 2 composants sont suffisants pour l'universalité des systèmes distribués à changement de phase, voir [22], puis qu'avec 2 composants il est possible de produire tous les langages récursivement énumérables, voir [21, 24]. Leur preuve se fondant sur la simulation d'une machine de Turing est strictement séquentielle : une seule molécule principale qui code le ruban et l'état de la machine doit être présente dans le système. Or, cela ne correspond pas à la nature parallèle des transformations biologiques où plusieurs évolutions se déroulent en même temps. Entre temps, une autre solution a été donnée par A. Păun dans [35] qui a montré qu'avec 4 composants il est possible de simuler le travail d'une grammaire arbitraire. Dans sa solution, plusieurs évolutions de molécules se font dans le même temps, sa variante est donc parallèle. Ses résultats ont été améliorés par M. Margenstern et Yu. Rogozhin qui ont montré qu'une grammaire arbitraire peut être simulée avec 3 composants, voir [23]. Les mêmes auteurs ont réussi à démontrer que les systèmes distribués à changement de phase avec un seul composant sont universels, voir [27], puis, qu'il est possible de produire tous les langages récursivement énumérables d'une manière séquentielle, voir [26]. Nous avons amélioré les résultats ci-dessus en montrant, en collaboration avec M. Margenstern et Yu. Rogozhin, qu'avec 2 composants il est possible de produire tous les langages récursivement énumérables d'une manière parallèle, voir [28]. Finalement, dans le cadre de la même collaboration, le point final a été mis en montrant qu'avec un seul composant, il est possible de produire tous les langages récursivement énumérables

d'une manière parallèle, voir [29]. Ces deux résultats peuvent être retrouvés aux chapitres 4 et 6 de cette thèse.

Nous ajoutons aussi qu'un logiciel de simulation des systèmes distribués à changement de phase développé par l'auteur dans le cadre de son DEA a été fortement utilisé pour la construction et pour la recherche d'erreurs dans les deux derniers systèmes. Ce logiciel a permis aussi de trouver des inexactitudes dans plusieurs articles concernant les systèmes distribués à changement de phase. Par exemple, les articles [23] et [44] nécessitent des petites corrections et la preuve donnée dans [35] doit être complètement révisée. Les détails sur ce sujet peuvent être trouvés au chapitre 4 du présent travail.

Une étude des systèmes distribués à changement de phase du point de vue de la calculabilité a été faite dans le mémoire de DEA de l'auteur [51] où des systèmes distribués à changement de phase pour les opérations arithmétiques de base (addition, multiplication, exponentiation et division) et la fonction d'Ackermann ont été explicitement donnés. Une réalisation de l'algorithme d'Euclide dans les systèmes distribués à changement de phase peut être trouvé sur le site web de l'auteur, voir [49], ainsi que le logiciel développé.

La grande puissance des systèmes distribués à changement de phase est obtenue grâce à leur définition et en particulier grâce au fait qu'une seule recombinaison est faite et que seul le résultat de cette recombinaison est gardé. D'un point de vue biologique il est plus logique de supposer que le nombre des recombinaisons n'est pas borné et que toutes les molécules qui peuvent prendre part dans une recombinaison ne sont pas éliminées. Ces observations nous conduisent aux systèmes distribués à changement de phase étendus introduits par M. Margenstern et Yu. Rogozhin dans [23]. Ces systèmes étant une combinaison à la fois des systèmes distribués à changement de phase et des systèmes de Head, il est possible de les regarder comme des systèmes distribués à changement de phase où chaque composant agit comme un système de Head. L'élimination est permise seulement dans le cas où la molécule considérée ne peut s'accorder avec aucune règle du composant courant, c'est-à-dire elle ne contient aucun site d'une règle de recombinaison du composant courant.

Maintenant un seul composant n'est plus suffisant pour obtenir une grande puissance d'expression et, dans ce cas, le langage produit est borné par la famille des langages rationnels. Mais deux composants suffisent pour produire tous les langages récursivement énumérables, fait démontré par M. Margenstern et Yu. Rogozhin dans [25]. Comme cette solution utilise une simulation des machines de Turing, c'est-à-dire une simulation séquentielle, le même problème de la production parallèle des langages récursivement énumérables peut être posé. Nous nous sommes intéressés à ce problème et nous avons trouvé progressivement des solutions, premièrement avec 4 composants, voir [50], puis avec 3, voir [52], et finalement avec 2, voir [54]. Pour ces systèmes, le problème de la production parallèle des langages récursivement énumérables est donc résolu. Les deux derniers résultats peuvent être trouvés aux chapitres 5 et 6.

2.6 Systèmes à membranes

Jusqu'à présent nous avons parlé des systèmes de Head, ainsi que de leurs extensions et modifications. Nous remarquons que l'inspiration pour ces modèles vient des procédés par lesquels les êtres vivants manipulent leur ADN. Une autre approche a été suivie par Gh. Păun qui, se plaçant au niveau de la cellule vivante et s'inspirant des processus intra et extracellulaires, a créé un nouveau paradigme de calcul : les systèmes à membranes ou systèmes P.

La cellule est considérée comme un ensemble de membranes imbriquées les unes dans les autres et pouvant contenir des objets et des règles d'évolution. L'arrangement spatial des membranes peut être représenté par un diagramme de Venn, voir figure 2.1, ou par un arbre dont les noeuds représentent les membranes et dont les arêtes sont définies par la relation « être contenu dans », voir figure 2.2. La même structure peut être définie à l'aide d'un mot du langage algébrique où les parenthèses correspondantes ont la même étiquette. Par exemple, la structure représentée à la figure 2.1 peut être décrite par le mot : $[_1\ [_2\ ]_2\ [_3\ ]_3\ [_4\ [_5\ ]_5\ [_6\ ]_6\ ]_4\ ]_1$.

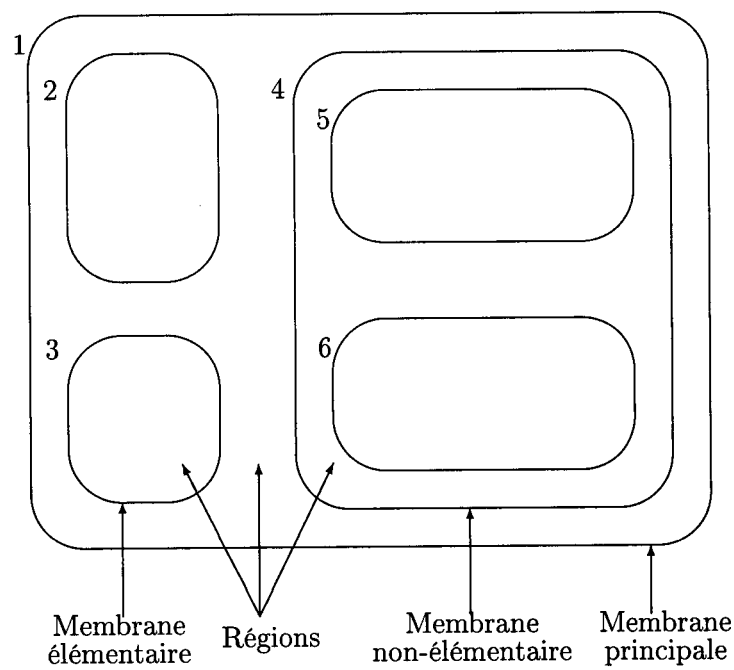


FIG. 2.1 – Une structure de membranes

Le modèle de base est très abstrait et il ne spécifie ni la nature des objets ni la nature des règles. De nombreuses variantes spécifient ces deux paramètres en obtenant une multitude de modèles de calcul. Il existe maintenant plus d'une

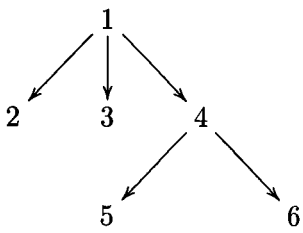


FIG. 2.2 – L'arbre représentant la structure des membranes de la figure 2.1

centaine de variantes et leur nombre croît constamment. Plus de détails peuvent être trouvés sur la page web [58] qui contient un excellent recueil de références sur les systèmes à membranes.

Les compartiments délimités par les membranes contiennent des multi-ensembles d'objets. La nature de ces objets peut être très variée. D'abord on peut considérer des objets non-structurés en faisant une analogie avec des substances chimiques à l'intérieur de la cellule. Puis, il est possible de structurer les objets et on peut les considérer comme des chaînes de caractères. Il existe même des variantes qui travaillent avec des objets compliqués comme des graphes, des images, etc. Davantage de précisions sur ces variantes peuvent être trouvées dans [43], ainsi que sur la page web [58].

Les règles qui sont généralement placées dans les membranes, présentent aussi une grande variété. Leur complexité dépend du type d'objets utilisés. Pour les objets simples, les règles transforment un multi-ensemble d'objets en un autre multi-ensemble d'objets. Pour les objets structurés comme des chaînes de caractères, des règles de réécriture ou de recombinaison sont utilisées.

Une propriété importante des systèmes à membranes est la communication. Toutes les règles contiennent des indicateurs de cible. Si la règle contient l'indicateur « out », le résultat de son application est envoyé dans la membrane immédiatement supérieure ; si la règle contient l'indicateur « here », le résultat reste dans la même membrane ; si l'indicateur « in » est présent, le résultat est envoyé dans une membrane intérieure choisie d'une manière non-déterministe.

Cette propriété de communication est tellement puissante qu'elle seule suffit pour obtenir une grande puissance d'expression, comme dans le cas des systèmes à membranes à communication de types symport/antiport. Ces derniers systèmes comprennent deux types de règles de communication : la règle de type symport, quand plusieurs objets passent ensemble d'une membrane à l'autre et la règle de type antiport, quand plusieurs objets de deux membranes sont échangés. Malgré la simplicité apparente, une membrane suffit pour simuler une machine à registres et, par conséquent, pour produire tous les ensembles récursivement énumérables.

Une autre variante des systèmes à membranes permet la modification de la structure des membranes par création ou suppression de membranes. Ces systèmes sont très intéressants parce qu'ils permettent de résoudre des problèmes NP-complets

dans un temps polynomial et même linéaire, voir [43, 58].

Nous nous intéressons à une classe particulière des systèmes à membranes : les systèmes à membranes avec recombinaison qui sont un mélange de systèmes à membranes et de systèmes de Head. Ces systèmes ont comme objets des chaînes de caractères et comme règles des règles de recombinaison enrichies par les indicateurs de cible. De plus, contrairement à la plupart des systèmes à membranes, on considère des ensembles de mots dans chaque membrane et non plus des multi-ensembles. Une étape de calcul représente une application parallèle des règles dans chaque membrane suivie de la communication des molécules résultantes selon les indicateurs de cible. Le résultat consiste en toutes les molécules faisant partie d'un alphabet terminal qui sont envoyées à l'extérieur de la membrane principale.

Ces systèmes ont été introduits par Gh. Păun dans [41] où leur universalité avec 4 membranes a été démontrée. Le nombre de membranes nécessaires pour produire tous les langages récursivement énumérables a été réduit à 3 dans [45], puis à 2 dans [36]. Le dernier résultat a été amélioré du point de vue de la taille des règles de recombinaison par P. Frisco dans [9].

Nous nous sommes intéressés aux systèmes à membranes avec recombinaison ayant une seule membrane. Nous avons trouvé que la définition initiale donné par Gh. Păun dans [41] n'est pas complète et qu'elle nécessite quelques précisions. Même la définition révisée apparue dans [43] ne comprend pas toutes les possibilités. Nous avons complété cette définition en proposant 5 variantes et nous avons montré que la puissance d'expression dépend de la variante utilisée. Nous avons montré que pour la variante la plus proche de celle proposée par Gh. Păun, la puissance d'expression est limitée à la famille des langages rationnels. Puis, nous avons montré qu'en considérant trois autres variantes, on obtient tous les langages récursivement énumérables. Pour la variante restante nous n'avons pas encore trouvé une description du langage produit, mais nous supposons qu'il est rationnel. Plus de détails peuvent être trouvés au chapitre 9.

Nous nous sommes intéressés aussi à la variante non-étendue des systèmes à membranes avec recombinaison. Dans ce cas il n'y a plus d'alphabet terminal et toutes les chaînes sorties à l'extérieur de la membrane principale sont considérées comme résultat. Gh. Păun dans [43] a montré comment à l'aide de 2 membranes additionnelles il est possible de transformer un système étendu en un système non-étendu. Comme il existe des systèmes étendus avec 2 membranes qui produisent tous les langages récursivement énumérables, la même puissance d'expression est obtenue avec 4 membranes dans le cas non-étendu. Nous avons montré qu'en arrangeant soigneusement les règles il est possible de parvenir à deux membranes seulement. Ce résultat ne peut plus être amélioré car avec une membrane dans le cas non-étendu on ne produit qu'un sous-ensemble des langages rationnels. Ce même résultat a été démontré dans [53] et peut être trouvé au chapitre 9.

Une autre variante des systèmes à membranes avec recombinaison, les systèmes à membranes avec recombinaison et communication immédiate, a été proposée par C. Martín-Vide, Gh. Păun et A. Rodríguez-Patón dans [30]. Dans cette variante, les résultats de l'application de chaque règle doivent quitter la membrane dans laquelle

ils ont été produits. En effet, c'est un cas particulier des systèmes à membranes avec recombinaison où toutes les règles possèdent les quatre combinaisons possibles des indicateurs de cible « in » et « out ». Ces systèmes sont capables de produire tous les langages récursivement énumérables, mais l'article original n'indiquait pas de nombre fixe de membranes. Nous avons trouvé des parallèles entre ces systèmes et les systèmes distribués à changement de phase et nous avons montré que 2 membranes suffisent pour produire tous les langages récursivement énumérables. Nous avons aussi montré qu'avec une seule membrane il n'est possible de produire que des langages finis, nous avons donc trouvé la frontière entre la décidabilité et l'indécidabilité pour ces systèmes. De même, nous avons trouvé une connexion avec les systèmes à membranes avec des règles de recombinaison placées sur des membranes et non plus dans les régions, voir [7].

Maintenant, nous pouvons passer à une analyse formelle de l'opération de recombinaison et des systèmes de Head.

Chapitre 3

La recombinaison simple par rapport à la recombinaison double

Dans ce chapitre nous introduisons les notions centrales de notre thèse : l'opération de la recombinaison et les systèmes de Head. Nous étudierons les deux définitions possibles de la recombinaison : la recombinaison simple et la recombinaison double, et nous montrons des classes de langages appartenant aux familles de langages des systèmes de Head fondés sur la recombinaison simple et la recombinaison double. Ces classes sont obtenues à partir des langages cL, Lc, LcL, cLc, où $L \subseteq V^*$ est un langage rationnel et où $c \notin V$. Nous montrons pour la première fois que certaines classes qui sont des langages de recombinaison simple, ne peuvent pas être des langages de recombinaison double. Cette dernière famille est donc strictement plus petite que l'autre. Nous montrons aussi des classes de langages rationnels qui ne sont pas des langages de recombinaison simple.

3.1 La définition formelle des systèmes de Head

Dans cette section nous donnerons la définition formelle des systèmes de Head ainsi que certains résultats connus.

On comprend par une *molécule* (abstraite) un mot sur un alphabet.

Définition 3.1.1. Une *règle de recombinaison* (sur un alphabet V) est un quadruplet (u_1, u_2, u_3, u_4) où $u_1, u_2, u_3, u_4 \in V^*$. On l'écrit souvent comme $u_1\#u_2\$u_3\#u_4$, $\{\$, \#\} \notin V$ ou encore en deux dimensions : $\begin{array}{c|c} u_1 & u_2 \\ \hline u_3 & u_4 \end{array}$. Les chaînes u_1u_2 et u_3u_4 sont appelés *sites* de recombinaison.

Nous disons qu'un mot x *s'accorde* avec la règle r s'il contient une occurrence d'un des deux sites de r. Nous disons aussi que x et y sont *complémentaires* par rapport à la règle r si x contient un site de r et y contient l'autre. Quand x et y avec $x = x_1u_1u_2x_2$ et $y = y_1u_3u_4y_2$ sont complémentaires par rapport à la règle $r = u_1\#u_2\$u_3\#u_4$ on peut définir l'application de r au couple x, y dont le résultat

est w et z où $w = x_1u_1u_4y_2$ et $z = y_1u_3u_2x_2$. On écrit ce fait de la manière suivante : $(x, y) \vdash_r (w, z)$. Dans ce cas, on dit qu'on effectue une *recombinaison double.* Si dans la définition précédente on considère seulement w comme résultat, on dit qu'on effectue une *recombinaison simple.* On l'écrit de la manière suivante : $(x, y) \vdash_r w$. Nous pouvons aussi écrire l'application d'une recombinaison double de la manière suivante :

$$\frac{x_1u_1 \mid u_2x_2}{y_1u_3 \mid u_4y_2} \vdash_r \frac{x_1u_1u_4y_2}{y_1u_3u_2x_2}.$$

D'une manière similaire nous pouvons écrire l'application d'une recombinaison simple.

Exemple 3.1.1.

On considère la règle r : $\frac{h \mid a}{b \mid i}$ et on l'applique aux molécules *chasse* et *bien* :

$$\frac{\mathbf{ch} \mid \mathbf{asse}}{\mathbf{b} \mid \mathbf{ien}} \vdash_r \frac{\mathbf{chien}}{\mathbf{basse}}.$$

La paire $\sigma = (V, R)$ où V est un alphabet et R est un ensemble de règles de recombinaison est appelée un *schéma de recombinaison* ou schéma.

Pour un schéma de recombinaison $\sigma = (V, R)$ et un langage formel $L \subseteq V^*$ on définit :

$\sigma_1(L) \stackrel{\text{def}}{=} \{w \in V^* | \exists x, y \in L, \exists r \in R : (x, y) \vdash_r w\}$.

$\sigma_2(L) \stackrel{\text{def}}{=} \{w, z \in V^* | \exists x, y \in L, \exists r \in R : (x, y) \vdash_r (w, z)\}$.

Maintenant, on peut introduire l'itération de l'opération de recombinaison.

$\sigma_j^0(L) = L$,

$\sigma_j^{i+1}(L) = \sigma_j^i(L) \cup \sigma_j(\sigma_j^i(L)),\ i \geq 0$,

$\sigma_j^*(L) = \cup_{i \geq 0} \sigma_j^i(L)$,

où $j \in \{1, 2\}$.

Nous montrons maintenant que l'opération de recombinaison préserve la rationalité d'un langage.

Théorème 3.1.1. *Soient $L \subseteq T^*$ un langage rationnel et $\sigma = (T, R)$ un schéma de recombinaison. Le langage $\mathcal{L} = \sigma_1(L)$ est aussi un langage rationnel.*

Démonstration. Nous allons construire une grammaire rationnelle qui produit $\mathcal{L}$. Pour obtenir cette grammaire, nous prenons une grammaire rationnelle G_1 qui produit L et nous transformons son ensemble de productions en ajoutant des productions rationnelles et en éliminant certaines productions initiales, ce qui permettra de simuler l'application de σ sur L.

Nous considérons donc la grammaire rationnelle $G_1 = (N_G, T, S, P_1)$ telle que $L(G_1) = L$. Nous ne détaillons les symboles de N_G, car ce n'est pas important pour notre démonstration. Nous pouvons considérer que toutes les règles de P_1 sont de la forme $A \to aB$ et $A \to a$, où $A, B \in N_G$ et $a \in T$. Nous pouvons supposer aussi que G_1 ne contient pas des productions avec ε et qu'elle ne possède pas des symboles inaccessibles.

Considérons aussi l'ensemble suivant des productions :

$$P_R = \left\{ U_S^i \to u_1^i u_4^i U_E^i : \exists r_i = \frac{u_1^i \mid u_2^i}{u_3^i \mid u_4^i} \in R \right\}.$$

P_1' : qui consiste en toutes les productions de P_1 où un prime est ajouté à chaque symbole non-terminal.

$$P_c = \left\{ \begin{array}{l} A \to U_S^i \\ U_E^i \to F' \ (F \neq \varepsilon) \\ U_E^i \to \varepsilon \ (F = \varepsilon) \end{array} : (A,B,C,D,E,F) \in ParseRule_i \right\}.$$

où l'ensemble $ParseRule_i$ est défini par :

$$ParseRule_i = \left\{ (A,B,C,D,E,F) : \exists A \underset{G_1}{\overset{*}{\Rightarrow}} u_1^i B \underset{G_1}{\overset{*}{\Rightarrow}} u_1^i u_2^i C,\ \exists D \underset{G_1}{\overset{*}{\Rightarrow}} u_3^i E \underset{G_1}{\overset{*}{\Rightarrow}} u_3^i u_4^i F, \right.$$

$$\left. \exists r_i = \frac{u_1^i \mid u_2^i}{u_3^i \mid u_4^i} \in R \right\}.$$

Nous observons que C et F peuvent être égaux à ε. Si $F = \varepsilon$, la deuxième production de P_c devient $U_E^i \to \varepsilon$.

En termes usuels, P_R contient les productions qui permettent de simuler l'application des règles $r_i \in R$, P_1' contient une copie des productions initiales et P_c fait la connexion entre P_1 et les deux ensembles ci-dessus.

Considérons maintenant $P = P_1 \setminus \{A \to a \in P_1\} \cup P_1' \cup P_R \cup P_c$.

Posons $G = (N,T,S,P)$, où $N = N_G \cup N_G' \cup \{U_S^i, U_E^i : \exists r_i \in R\}$. Il est facile de voir que G est une grammaire rationnelle. Les règles de P_1 et P_1' sont déjà de la bonne forme, tandis que les règles de P_R et P_c peuvent être facilement transformées en forme nécessaire en utilisant des procédés standard comme l'élimination des renommages et des productions vides.

Nous observons que pour obtenir un mot dans G nous devons utiliser soit une production de P_1', soit une production de la forme $U_E^i \to \varepsilon$ de P_c, car nous avons éliminé de P toutes les productions terminales de P_1.

La figure 3.1 illustre comment la grammaire G_1 est transformé en G. Nous substituons deux dérivations dans G_1, indiquées par une ligne en pointillés, par une dérivation dans G, indiquée par une ligne continue, qui correspond à la recombinaison des mots initiaux. De même, les dérivations initiales deviennent inaccessibles dans G.

Nous affirmons que $L(G) = \sigma_1(L)$.

La figure 3.1 illustre aussi l'idée de la démonstration. Une ligne continue représente une dérivation dans G, tandis que les lignes en pointillés représentent des dérivations dans G_1. Pour une inclusion, nous devons construire une dérivation indiquée par la ligne continue à partir des deux dérivations indiquées par la ligne en pointillés, tandis que pour l'autre inclusion nous reconstruisons les deux dérivations indiquées par la ligne en pointillés à partir de la dérivation indiquée par la ligne continue. De même, la règle de R appliquée peut être reconstruite à partir de ces dérivations.

1. $\mathcal{L} \subseteq L(G)$.

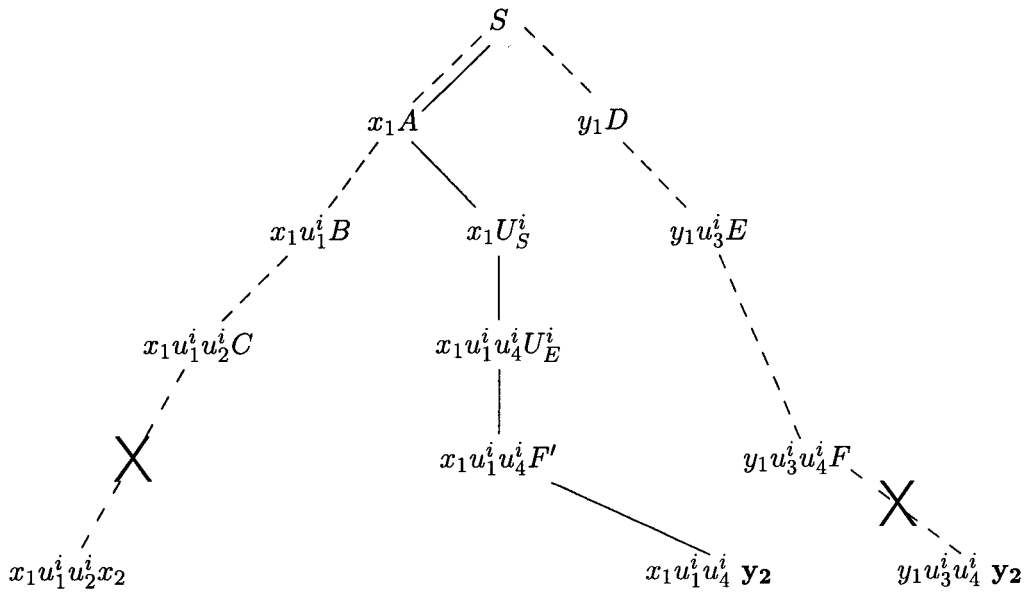


FIG. 3.1 – Transformation de G_1 dans G

Soit w dans $\mathcal{L}$. Dans ce cas $w \in T^*$ et il existe des mots $x_1u_1^iu_2^ix_2$ et $y_1u_3^iu_4^iy_2$ dans $\mathcal{L}$ et une règle $\begin{array}{c|c} u_1^i & u_2^i \\ \hline u_3^i & u_4^i \end{array}$ dans R tels que $(x,y) \vdash_r w = x_1u_1^iu_4^iy_2$.

Comme $x_1u_1^iu_2^ix_2$ est dans $\mathcal{L}$, nous avons la dérivation suivante dans G_1 :
$S \underset{P_1}{\overset{*}{\Rightarrow}} x_1A \underset{P_1}{\overset{*}{\Rightarrow}} x_1u_1^iB \underset{P_1}{\overset{*}{\Rightarrow}} x_1u_1^iu_2^iC \underset{P_1}{\overset{*}{\Rightarrow}} x_1u_1^iu_2^ix_2$.

D'une manière similaire $y_1u_3^iu_4^iy_2 \in \mathcal{L}$ implique qu'il existe la dérivation suivante :
$S \underset{P_1}{\overset{*}{\Rightarrow}} y_1D \underset{P_1}{\overset{*}{\Rightarrow}} y_1u_3^iE \underset{P_1}{\overset{*}{\Rightarrow}} y_1u_3^iu_4^iF \underset{P_1}{\overset{*}{\Rightarrow}} y_1u_3^iu_4^iy_2$.

Il existe alors une dérivation dans G :
$S \underset{P_1}{\overset{*}{\Rightarrow}} x_1A \underset{P_c}{\overset{1}{\Rightarrow}} x_1U_S^i \underset{P_R}{\overset{1}{\Rightarrow}} x_1u_1^iu_4^iU_E^i \underset{P_c}{\overset{1}{\Rightarrow}} x_1u_1^iu_4^iF' \underset{P_1'}{\overset{*}{\Rightarrow}} x_1u_1^iu_4^iy_2$, car si $F \underset{P_1}{\overset{*}{\Rightarrow}} y_2$, alors $F' \underset{P_1'}{\overset{*}{\Rightarrow}} y_2$.

Nous avons aussi $x_1u_1^iu_4^iy_2 \in T^*$, par conséquent, $w \in L(G)$.

2. $L(G) \subseteq \mathcal{L}$.

Nous observons qu'il est impossible de produire un mot terminal dans G sans avoir utilisé des productions de P_1', ou une des productions $U_E^i \to \varepsilon$ de P_c. De même, pour arriver à un symbole non-terminal ayant un prime, ou à U_E^i, nous devons utiliser des productions de P_R et P_c.

Nous rappelons aussi que l'idée de la démonstration est de transformer une dérivation dans G en deux dérivations dans G_1 qui produiront les deux mots auxquels une règle de recombinaison a été appliquée.

Soit w dans $L(G)$. Il existe alors une règle $U_E^i \to F'$ dans P_c (le cas de la règle

$U_E^i \to \varepsilon \in P_c$ est similaire) telle que l'on ait la dérivation suivante de w :
$S \underset{P_1}{\overset{*}{\Rightarrow}} x_1 A \underset{P_c}{\overset{1}{\Rightarrow}} x_1 U_S^i \underset{P_R}{\overset{1}{\Rightarrow}} x_1 u_1^i u_4^i U_E^i \underset{P_c}{\overset{1}{\Rightarrow}} x_1 u_1^i u_4^i F' \underset{P_1'}{\overset{*}{\Rightarrow}} x_1 u_1^i u_4^i y_2 = w.$

Par conséquent, il existe une règle $r_i = \dfrac{u_1^i \mid u_2^i}{u_3^i \mid u_4^i}$ dans R et on a les dérivations suivantes dans G_1 :

$S \underset{P_1}{\overset{*}{\Rightarrow}} x_1 A \underset{P_1}{\overset{*}{\Rightarrow}} x_1 u_1^i B \underset{P_1}{\overset{*}{\Rightarrow}} x_1 u_1^i u_2^i C \underset{P_1}{\overset{*}{\Rightarrow}} x_1 u_1^i u_2^i x_2.$

$S \underset{P_1}{\overset{*}{\Rightarrow}} y_1 D \underset{P_1}{\overset{*}{\Rightarrow}} y_1 u_3^i E \underset{P_1}{\overset{*}{\Rightarrow}} y_1 u_3^i u_4^i F \underset{P_1}{\overset{*}{\Rightarrow}} y_1 u_3^i u_4^i y_2.$

L'existence de A, B, C, D, E, F est garantie par la définition de P_c. De même, nous pouvons produire le deuxième mot, car D est accessible depuis l'axiome.

Nous obtenons donc $x = x_1 u_1^i u_2^i x_2$ et $y = y_1 u_3^i u_4^i y_2$ qui font partie de $L(G_1)$. Dans ce cas x et y sont dans L.

Nous pouvons donc appliquer r_i à x et y : $\dfrac{x_1 u_1^i \mid u_2^i x_2}{y_1 u_3^i \mid u_4^i y_2} \vdash_{r_i} x_1 u_1^i u_4^i y_2.$

Comme $w \in T^*$, nous obtenons que w est dans $\mathcal{L}$.

□

Si nous itérons le processus de construction de G à partir de G_1 en utilisant à chaque fois la grammaire nouvelle obtenue à la place de G_1 et sans éliminer les productions de P_1, on obtient une grammaire qui simule l'itération de σ sur L. La preuve de cette affirmation n'est pas triviale et elle peut être retrouvée dans [44], où ce résultat est démontré en termes d'automates finis.

Définition 3.1.2. [12, 14] Un *système de Head* est définit par la donnée du couple $H = (\sigma, A) = ((V, R), A)$, écrit plus simplement $H = (V, A, R)$, où V est un alphabet fini, $A \subseteq V^*$ est un ensemble de mots initiaux, appelés axiomes, et $R \subseteq V^* \times V^* \times V^* \times V^*$ est un ensemble de règles de recombinaison.

Le système de Head H est dit fini si A et R sont des ensembles finis.

Le *langage produit* par le système de Head H fondé sur la recombinaison double, respectivement simple, est $L(H) \overset{\text{def}}{=} \sigma_2^*(A)$, respectivement $L(H) \overset{\text{def}}{=} \sigma_1^*(A)$.

Donc, le langage produit par le système de Head H est l'ensemble de toutes les molécules produites par l'application itérative des règles de R aux copies des molécules obtenues précédemment et en partant de A comme ensemble de molécules initiales.

Exemple 3.1.2.

Dans cet exemple on utilise la recombinaison double.

On considère le système $H = ((V, R), A)$ ou $V = \{a, b\}$, $R = \left\{ r : \dfrac{a \mid b}{a \mid a} \right\}$ et $A = \{abaa\}$.

On commence avec $abaa$. Donc, $\sigma_2^0(A) = \{abaa\}$. On peut appliquer la règle r à cette molécule et à elle-même : $(a|baa, aba|a) \vdash_r (aa, ababaa)$, où on a souligné par | les sites de recombinaison. On obtient $\sigma_2^1(A) = \{abaa, aa, ababaa\}$. A l'étape suivante on a les applications suivantes :

$(a|baa, a|a) \vdash_r (aa, abaa)$,

$(a|baa, ababa|a) \vdash_r (aa, abababaa)$,
$(a|babaa, a|a) \vdash_r (aa, ababaa)$,
$(a|babaa, aba|a) \vdash_r (aa, abababaa)$,
$(a|babaa, ababa|a) \vdash_r (aa, ababababaa)$.
Cela donne $\sigma_2^2(A) = \{abaa, aa, ababaa, abababaa, ababababaa\}$

On voit que chaque recombinaison ajoute des $(ab)^*$ à gauche de la molécule existante. Le langage produit par ce système est donc $L(H) = (ab)^*aa$.

Maintenant, on considère le même système où on prend $A = \{aab\}$. Le fonctionnement du nouveau système H' est différent. Premièrement on applique aab à elle-même par la règle r : $(aa|b, a|ab) \vdash_r (aaab, ab)$. Puis on a les possibilités suivantes :
$(a|b, a|ab) \vdash_r (aab, ab)$,
$(a|b, a|aab) \vdash_r (aaab, ab)$,
$(aa|b, a|aab) \vdash_r (aaaab, ab)$,
$(aaa|b, a|aab) \vdash_r (aaaaab, ab)$,
On voit facilement qu'on ajoute des a à gauche des molécules existantes. Le langage produit par H' est donc $L(H') = a^+b$.

Remarque 3.1.1. Soient $S = (V, A, R)$ et $S' = (V, A, R')$ deux systèmes de Head et $R \subseteq R'$. On a $L(S) \subseteq L(S')$.

Nous notons par $H_1(FIN, FIN)$ la famille des langages produits par des systèmes de Head finis fondés sur la recombinaison simple. Pareillement, nous notons par $H_2(FIN, FIN)$ la famille des langages produits par des systèmes de Head finis fondés sur la recombinaison double. Le langage produit par un système de Head fondé sur la recombinaison simple, respectivement double, est appelé un *langage de recombinaison* simple, respectivement double.

Dans le reste de ce chapitre nous considérons uniquement des langages rationnels non-finis et des systèmes de Head finis.

Proposition 3.1.2. *[14, 44]* $H_2(FIN, FIN) \subseteq H_1(FIN, FIN)$.

Cette proposition se fonde sur le raisonnement suivant. Un système de Head est appelé symétrique si pour chaque règle $r = u_1\#u_2\$u_3\#u_4 \in R$, R contient la règle $r' = u_3\#u_4\$u_1\#u_2$. Il est facile d'observer qu'un système de Head fondé sur la recombinaison double est implicitement supposé symétrique. Par conséquent, chaque langage produit par un système dans $H_2(FIN, FIN)$ peut être produit par un système dans $H_1(FIN, FIN)$ qui est symétrique. Cet argument a été utilisé dans [14].

Cette propriété est une des raisons pour lesquelles la recombinaison double est généralement utilisée dans la littérature, car tous les résultats obtenus pour la recombinaison double peuvent être facilement traduits en termes de recombinaison simple.

Maintenant nous montrons les limites des systèmes de recombinaison finis.

Théorème 3.1.3. *[33, 14, 44]* $H_1(FIN, FIN) \subseteq REG$.

Le résultat énoncé reste vrai même si un ensemble rationnel d'axiomes est utilisé. La preuve de ce résultat peut être trouvée dans [33] et [14]. L'ouvrage [44] contient

une preuve de ce résultat fondée sur la simulation de l'opération de la recombinaison par un automate fini, laquelle est beaucoup plus simple que les deux précédentes. En effet, la démonstration utilise une construction similaire à celle utilisée pendant la preuve du théorème 3.1.1, mais traduite en termes d'automates finis.

Maintenant, nous montrons que l'inclusion précédente est stricte.

Proposition 3.1.4. $L_{aa} = (aa)^* \in REG \setminus H_1(FIN, FIN)$.

Démonstration. Nous procédons par contradiction. Soit $S = (V, A, R)$ un système de Head qui produit L_{aa}. On a $V = \{a\}$. Soit $r \in R$, $r = a^k\#a^m\$a^p\#a^q$. Considérons un entier $i \geq 1$ tel que $k + m + i > p + q$ et que $k + m + i$ est pair. Considérons maintenant $x = a^{k+m+i}$. Par hypothèse $x \in L(S)$. Considérons maintenant deux cas selon la parité de $k + q$.

1. $k+q$ est pair. On a alors l'application suivante de la règle r à x et à lui-même. $(aa^k|a^m a^{i-1}, a^{k+m+i-(p+q)}a^p|a^q) \vdash_r a^{k+q+1} \notin L_{aa}$.
   Ceci contredit l'hypothèse, car S est fermé par rapport à la recombinaison.
2. $k+q$ est impair. On a alors l'application suivante de la règle r à x et à lui-même. $(a^k|a^m a^i, a^{k+m+i-(p+q)}a^p|a^q) \vdash_r a^{k+q} \notin L_{aa}$.
   Ce qui est une contradiction.

□

On obtient donc le résultat suivant.

Théorème 3.1.5. $H_1(FIN, FIN) \subset REG$.

3.2 Exemples des classes de langages de recombinaison

Dans ce qui suit nous nous concentrons sur des langages obtenus par union à partir d'un langage rationnel $L \subseteq V^*$ et des langages Lc, cL, LcL et cLc, où $c \notin V$, c'est-à-dire c est une constante pour ces langages. Le théorème suivant est une conséquence d'un résultat plus général énoncé par T. Head dans [13]. Il montre que parmi les langages ci-dessus, certains sont des langages de recombinaison.

Théorème 3.2.1. *[13] Soit $L \subseteq V^*$ un langage rationnel et $c, d \notin V$. Alors, les langages Lc, cL, LcL, cLc et $cL + Ld$ sont dans $H_2(FIN, FIN)$. De plus, chaque règle du système de Head produisant Lc, respectivement cL, LcL, cLc et $cL + Ld$, est de la forme $m\#\varepsilon\$m'\#\varepsilon$ ou $\varepsilon\#m\$\varepsilon\#m'$, où m et m' sont des constantes pour Lc, respectivement cL, LcL, cLc et $cL + Ld$, et où m et m' contiennent aussi c dans le cas des langages Lc, cL, LcL et cLc.*

Remarque 3.2.1. Nous remarquons que dans [13], l'auteur a utilisé la recombinaison simple. Mais, comme il a été montré dans [2, 3], ce résultat reste valide pour le cas de la recombinaison double.

3.2.1 Langages de recombinaison double

Maintenant, nous présentons quelques classes de langages de recombinaison double obtenus à partir des langages ci-dessus.

Proposition 3.2.2. *Soit $L \subseteq V^*$ un langage rationnel et $c \notin V$. Les langages rationnels $\mathcal{L} = L + Lc^*$ et $\mathcal{L}' = L + c^*L$ sont des langages de recombinaison double.*

Démonstration. Considérons le langage $\mathcal{L}$. En vertu du théorème 3.2.1 nous obtenons qu'il existe un système de Head fondé sur la recombinaison double $S = (V \cup \{c\}, A, R)$ qui produit Lc. Soit $\bar{S} = (V \cup \{c\}, A, R \cup \{r\})$, où $r = \varepsilon\#c\$c\#\varepsilon$. Il est clair que $L(S) \subseteq L(\bar{S})$, voir remarque 3.1.1.

Il est facile d'observer que $L(\bar{S}) = \mathcal{L}$. Si on applique r à lc et lc, $l \in L$, on obtient l et lcc. Si on applique encore une fois r à lc et lcc, on obtient $lccc$, etc.

Plus rigoureusement, nous allons montrer d'abord que $\mathcal{L} \subseteq L(\bar{S})$, par récurrence. Soit w dans $\mathcal{L}$. Si w est dans Lc, on obtient que w est dans $L(S)$ qui est un sous-ensemble de $L(\bar{S})$. Si w est dans L, le mot wc se trouve dans Lc et, par conséquent, on peut le produire dans $\bar{S}$. Dans ce cas nous pouvons appliquer r à wc et à lui-même : $(wc, wc) \vdash_r (w, wcc)$ en obtenant w.

Maintenant, supposons que w est dans Lc^i, $i > 0$, c'est-à-dire que $w = w'c^i$, où $w' \in L$. Montrons que $w'c^{i+1} \in L(\bar{S})$. Comme wc et wc^i sont dans $L(\bar{S})$ par hypothèse de récurrence, nous obtenons wc^{i+1} en appliquant r à ces deux mots : $(wc, wc^{i-1}) \vdash_r (w, wc^i)$.

Inversement, soit w dans $L(\bar{S})$. Nous utilisons la récurrence sur le nombre k d'itérations de σ utilisées pour produire w. Si $k = 0$, le mot w est dans A qui fait partie de $L(S)$ qui est égal à Lc. Considérons maintenant un mot w qui est produit en $k+1$ itérations de σ. Soit la dernière application de σ : $(x, y) \vdash_{r'} (w_1, w_2)$ et $w \in \{w_1, w_2\}$. Par hypothèse de récurrence, x et y sont dans $\mathcal{L}$. Si r' est une règle de R, on obtient que x et y sont dans Lc^+, car les sites de r' contiennent c, voir théorème 3.2.1 et, par conséquent, w est aussi dans Lc^+. Si $r' = r$, nous obtenons que $x = x_1x_2$, avec $x_1 \in Lc^*$ et $x_2 \in c^+$ et $y = y_1y_2$, avec $y_1 \in Lc^+$ et $y_2 \in c^*$. Dans ce cas, $w_1 = x_1y_2$ fait partie de Lc^* et $w_2 = y_1x_2$ fait partie de Lc^+, c'est-à-dire que w est dans $\mathcal{L}$.

Il est possible de montrer d'une manière similaire que $\mathcal{L}'$ est un langage de recombinaison double. □

Proposition 3.2.3. *Soit $L \subseteq V^*$ un langage rationnel et $c \notin V$. Les langages rationnels $\mathcal{L} = L + Lc$ et $\mathcal{L}' = L + cL$ sont des langages de recombinaison simple.*

Démonstration. Nous utilisons la même construction que dans la proposition précédente.

Comme $c \notin V$, c est une constante pour Lc. Il existe donc un système de recombinaison simple $S = (V \cup \{c\}, A, R)$ tel que $Lc = L(S)$ (théorème 3.2.1). D'une manière similaire à la proposition 3.2.2 nous considérons la règle $r = \varepsilon\#c\$c\#\varepsilon$ et le système $\bar{S} = (V \cup \{c\}, A, R \cup \{r\})$. Il est clair que $L + Lc = L(\bar{S})$, car les règles de R permettent de produire Lc, tandis que r permet de produire L. Nous observons qu'il est important que la recombinaison simple soit utilisée. □

Nous ne savons pas si ce résultat reste vrai si on utilise la recombinaison double, mais nous pensons que c'est le cas.

Conjecture 3.2.1. Soit $L \subseteq V^*$ un langage rationnel et $c \notin V$. Les langages rationnels $\mathcal{L} = L + Lc$ et $\mathcal{L}' = L + cL$ sont des langages de recombinaison double.

Nous montrons maintenant une possibilité de résolution de cette conjecture. Supposons qu'il soit possible de trouver un entier $n > 0$ et des mots $z_i \in V^*$ et $w_i \in L$, $1 \leq i \leq n$ tels que pour tout $xw_iy \in L$ les deux conditions suivantes soient satisfaites :

a) $xw_i \in L$

b) $\forall w \in L : w = w'z_i \Rightarrow wy = w'z_iy \in L$

S'il est possible de trouver de tels z_i et w_i, alors, en ajoutant les mots w_i aux axiomes, les règles $z_i\#c\$w_i\#\varepsilon$ permettent de produire L à partir de Lc. Des conditions similaires peuvent être formulées pour produire L à partir de cL.

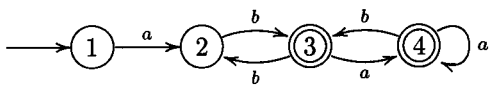
La motivation de l'énoncé précédent est la suivante. Tout d'abord, nous observons que nous pouvons produire le langage Lc, voir théorème 3.2.1. Puis, l'idée principale est de produire le mot x de L à partir du mot $xc \in Lc$ en effaçant le c à la fin. Ceci peut être obtenu si on utilise les règles $z_i\#c\$w_i\#\varepsilon$. Les conditions a) et b) garantissent la cohérence des mots obtenus, car les deux mots résultants doivent appartenir à L où à Lc.

Exemple 3.2.1.

Les langages $a^* + a^*b$, $a^* + ba^*$ sont des exemples qui confirment la conjecture ci-dessus. Pour le premier langage il suffit de prendre $i = 1$, $w_1 = a$ et $z_1 = \varepsilon$ pour satisfaire les deux conditions ci-dessus. La règle $\varepsilon\#b\$a\#\varepsilon$ permet alors de produire a^* à partir de a^*b. Nous observons que le langage a^*b peut être produit par un système de recombinaison double (théorème 3.2.1).

Exemple 3.2.2.

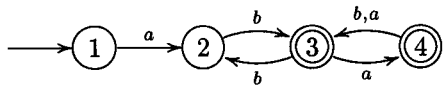
Considérons le langage rationnel L reconnu par l'automate fini suivant :



Si nous prenons $i = 2$ et $z_1 = a$, $z_2 = b$, $w_1 = ababa$, $w_2 = abbb$, les deux conditions ci-dessus sont satisfaites. Dans ce cas, les règles $a\#c\$ababa\#\varepsilon$ et $b\#c\$abbb\#\varepsilon$ permettent de produire L à partir de Lc. Nous observons que Lc peut être produit par un système de recombinaison double (théorème 3.2.1).

Exemple 3.2.3.

Considérons le langage rationnel L reconnu par l'automate fini suivant :



Dans ce cas, $i = 4$ et les 4 règles suivantes $ba\#c\$ababa\#\varepsilon$, $aa\#c\$abbb\#\varepsilon$, $ab\#c\$abbb\#\varepsilon$ et $bb\#c\$abbb\#\varepsilon$ satisfont les deux conditions ci-dessus. Elles permettent donc de produire L à partir de Lc. Nous observons que Lc peut être produit par un système de recombinaison double (théorème 3.2.1).

3.2.2 Classes de langages de recombinaison simple qui ne sont pas des langages de recombinaison double

La proposition suivante montre une classe de langages rationnels qui ne sont pas des langages de recombinaison double.

Proposition 3.2.4. *Soit $L \subseteq V^*$ un langage rationnel et $c, d \notin V$. Le langage $\mathcal{L}$ défini par $\mathcal{L} = L + cL + Ld$ n'est pas un langage de recombinaison double.*

Démonstration. Nous allons prouver ce résultat par contradiction. Nous supposons qu'il existe un système de recombinaison double $S = (V, A, R)$ tel que $L(S) = \mathcal{L}$. Nous considérons chaque règle $r \in R$ et nous montrons qu'en utilisant cette règle nous obtenons soit un mot ne faisant pas partie de $\mathcal{L}$ parce qu'il contient deux occurrences de lettres de V', où $V' = \{c, d\}$, soit les deux mots résultants contiennent une occurrence de c ou d, c'est-à-dire que nous ne pouvons pas produire des mots de L en utilisant cette règle. Nous remarquons aussi la propriété de fermeture de $\mathcal{L}$: $\mathcal{L} = \sigma_2(\mathcal{L})$ où σ est le schéma de recombinaison (V, R).

Considérons donc une règle $r = u_1\#u_2\$u_3\#u_4 \in R$ et soient $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$, $w_1 = x_1u_1u_4y_2$, $w_2 = y_1u_3u_2x_2$ tels que nous pouvons avoir l'application suivante de r : $(x, y) \vdash_r (w_1, w_2)$.

Il est clair que $|u_1u_2|_{V'} \leq 1$ et $|u_3u_4|_{V'} \leq 1$. On a donc les cas suivants quant au nombre de c et d dans les composants u_i de la règle, $1 \leq i \leq 4$:

1. $|u_i|_{V'} = 0$. Nous obtenons une contradiction si nous prenons x dans cL et y dans Ld, car w_1 contiendra c et d à la fois.
2. $|u_1u_2|_{V'} = 0$ et $u_3u_4 \in Fact(cL)$. Nous avons deux sous-cas possibles :
   (a) $u_3 \neq \varepsilon$. Si nous prenons x dans Ld et y dans cL, nous obtenons que $|w_2|_{V'} = 2$, ce qui est une contradiction.
   (b) $u_3 = \varepsilon$. Si nous prenons x et y dans cL nous obtenons une contradiction, car le mot résultant w_1 contiendra deux lettres c.
3. Nous pouvons raisonner d'une manière similaire pour les autres combinaisons où l'un des sites ne contient aucune lettre de V' et l'autre en contient une seule.
4. $u_1u_2 \in Fact(cL)$ et $u_3u_4 \in Fact(cL)$. Nous avons 3 sous-cas :
   (a) $u_3 = \varepsilon$ et $u_1 \neq \varepsilon$ (ou $u_1 = \varepsilon$ et $u_3 \neq \varepsilon$). Si nous prenons x et y dans cL, nous obtenons un mot résultant contenant deux occurrences de c.
   (b) $u_1, u_3 = \varepsilon$. Les deux mots résultants sont identiques aux mots initiaux.
   (c) $u_1, u_3 \neq \varepsilon$. À nouveau, les deux mots résultants font partie de cL.
   On en déduit qu'en utilisant des règles de ces deux derniers types nous ne pouvons pas produire de mot dans L.
5. Nous pouvons raisonner d'une manière similaire pour le cas où $u_1u_2 \in Fact(Ld)$ et $u_3u_4 \in Fact(Ld)$.

6. $u_1u_2 \in Fact(cL)$ et $u_3u_4 \in Fact(Ld)$. Nous avons 4 sous-cas :
   (a) $u_1, u_4 \neq \varepsilon$. Nous obtenons une contradiction si nous prenons x dans cL et y dans Ld, car w_1 contiendra c et d à la fois.
   (b) $u_1 \neq \varepsilon, u_4 = \varepsilon$. On obtient que $|w_1|_c = 1$ et $|w_2|_d = 1$.
   (c) $u_1 = \varepsilon, u_4 \neq \varepsilon$. On obtient que $|w_1|_d = 1$ et $|w_2|_c = 1$.
   On voit qu'en utilisant des règles de ces deux derniers types nous ne pouvons pas produire de mot dans L.
   (d) $u_1, u_4 = \varepsilon$. Nous obtenons une contradiction en prenant x dans cL et y dans Ld, car le mot résultant w_2 contiendra c et d à la fois.

En résumé, nous avons des règles de deux types. Les règles du premier type produisent des mots ayant une occurrence de c ou d, ces mots ne peuvent donc pas appartenir à L. Si nous supposons que nous pouvons produire L en utilisant les règles de l'autre type, nous pouvons alors montrer pour chaque règle de cette sorte un exemple de mots tels qu'en leur appliquant cette règle un des mots résultants contiendra deux occurrences de lettres de V', de sorte qu'on aura un mot qui ne fait pas partie de $\mathcal{L}$. Ce qui contredit le fait que $\mathcal{L}$ est fermé par rapport à la recombinaison. Par conséquent, nous obtenons une contradiction avec le fait que $\mathcal{L} = L + cL + Ld$ est un langage de recombinaison double. □

Nous pouvons facilement obtenir le même résultat en posant $|V'| = 1$, ce qui signifie que $c = d$.

Corollaire 3.2.5. *Soit $L \subseteq V^*$ un langage rationnel et $c \notin V$. Le langage $\mathcal{L}'$ défini par $\mathcal{L}' = L + cL + Lc$ n'est pas un langage de recombinaison double.*

Les théorèmes suivants montrent que les langages ci-dessus appartient à la famille $H_1(FIN, FIN)$. Ils sont donc dans $H_1(FIN, FIN) \setminus H_2(FIN, FIN)$.

Théorème 3.2.6. *Soit $L \subseteq V^*$ un langage rationnel et $c, d \notin V$. Le langage $\mathcal{L}$ défini par $\mathcal{L} = L + cL + Ld$ est dans $H_1(FIN, FIN) \setminus H_2(FIN, FIN)$.*

Démonstration. La proposition 3.2.4 nous donne que $\mathcal{L}$ défini par $\mathcal{L} = L + cL + Ld$ n'est pas dans $H_2(FIN, FIN)$. Il suffit maintenant de montrer que $\mathcal{L}$ est dans $H_1(FIN, FIN)$. Comme c est une constante pour cL et comme d est une constante pour Ld, cL et Ld sont des langages de recombinaison double, voir théorème 3.2.1 et, par conséquent, des langages de recombinaison simple, voir proposition 3.1.2. Soient $S_c = (V \cup \{c\}, A_c, R_c)$ le système de Head qui produit cL, c'est-à-dire $L(S_c) = cL$, et $S_d = (V \cup \{d\}, A_d, R_d)$ le système de Head qui produit Ld, c'est-à-dire $L(S_d) = Ld$. Le langage $cL+Ld$ est aussi un langage de recombinaison double, voir théorème 3.2.1. Nous affirmons que le système $S = (V \cup \{c, d\}, A_c \cup A_d, R_c \cup R_d \cup \{r\})$, où $r = \varepsilon\#d\$d\#\varepsilon$, produit $\mathcal{L}$, c'est-à-dire $\mathcal{L} = L(S)$. Nous observons que nous pouvons aussi utiliser la règle $r = \varepsilon\#c\$c\#\varepsilon$. En effet, nous pouvons produire cL et Ld en utilisant les règles de R_c et R_d. Finalement, la règle $r = \varepsilon\#d\$d\#\varepsilon$ qui s'applique uniquement à une paire de mots dans Ld, permet de produire les mots de L par

une recombinaison simple. Nous remarquons que le système S ainsi obtenu est un système de recombinaison simple, car la règle r ne peut pas être utilisée pour une recombinaison double. □

Le résultat précédent reste valide pour $\mathcal{L}' = L + cL + Lc$. Plus exactement, nous observons que la preuve que $\mathcal{L} = L + cL + Ld$ est un langage de recombinaison simple utilise l'hypothèse que c et d sont deux constantes pour $\mathcal{L}$. D'autre part il est possible de montrer que $\mathcal{L}' = L + cL + Lc$ est un langage de recombinaison simple, car $\{ca \mid a \in V\}$ et $\{ac \mid a \in V\}$ sont deux ensembles finis de constantes pour $\mathcal{L}'$. Dans ce cas, pour produire L nous pouvons utiliser l'ensemble de règles $r_a = a\#c\$ac\#\varepsilon$, respectivement $r_a = \varepsilon\#ca\$c\#a$, pour tout $a \in V$. Une telle règle r_a s'applique à deux mots de Lc, respectivement cL, produisant ainsi un mot de L.

Théorème 3.2.7. *Soit $L \subseteq V^*$ un langage rationnel et $c \notin V$. Le langage rationnel $\mathcal{L} = L + LcL$ n'est pas un langage de recombinaison double.*

Démonstration. Nous allons prouver ce résultat par contradiction. Nous supposons qu'il existe un système de recombinaison double $S = (V, A, R)$ tel que $L(S) = \mathcal{L}$. Nous considérons chaque règle $r \in R$ et nous montrons qu'en utilisant cette règle nous obtenons soit un mot ne faisant pas partie de $\mathcal{L}$ parce qu'il contient deux occurrences de c, soit deux mots qui contiennent c. Cela veut dire qu'il n'est pas possible de produire des mots de L en utilisant cette règle. Nous remarquons aussi la propriété de fermeture de $\mathcal{L}$: $\mathcal{L} = \sigma_2(\mathcal{L})$ où σ est le schéma de recombinaison (V, R).

Considérons donc une règle $r = u_1\#u_2\$u_3\#u_4 \in R$ et soient $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2$, $w_1 = x_1u_1u_4y_2$, $w_2 = y_1u_3u_2x_2$ tels que nous pouvons avoir l'application suivante de r : $(x, y) \vdash_r (w_1, w_2)$.

Il est clair que $|u_1u_2|_c \leq 1$ et $|u_3u_4|_c \leq 1$. On a donc les cas suivants quant au nombre de c dans les composants u_i de la règle, $1 \leq i \leq 4$:

1. $|u_i|_c = 0, i = 1, \ldots, 4$.
   Nous obtenons une contradiction si nous prenons x, y tels que $|x_1|_c = |y_2|_c = 1$, car alors $|w_1|_c = 2$.
   Nous affirmons qu'il est possible de trouver de tels x et y du fait de la définition de $\mathcal{L}$. C'est-à-dire si nous pouvons appliquer r à x et à y avec $|x_1|_c = 0$, nécessairement, $x = x_1u_1u_2w'cw''$. Comme $x_1u_1u_2w' \in L$, il existe un mot $x' = w_3cx_1u_1u_2w' \in LcL$ pour lequel $|x'_1|_c = |w_3cx_1|_c = 1$ et tel que la règle r puisse être appliquée à x' et à y.
2. $|u_1|_c = |u_2|_c = 0$ et $|u_3u_4|_c = 1$
   (a) $|u_3|_c = 1$ et donc $|u_4|_c = 0$: si nous prenons x tel que $|x_2|_c = 1$, nous obtenons une contradiction, car alors $|w_2|_c = 2$.
   (b) $|u_3|_c = 0$ et donc $|u_4|_c = 1$: si nous prenons x tel que $|x_1|_c = 1$, nous obtenons de même une contradiction, car cette fois $|w_1|_c = 2$.

2'. D'une manière similaire nous obtenons une contradiction dans le cas où l'on a $|u_3|_c = |u_4|_c = 0$ et donc $|u_1u_2|_c = 1$.

3. $|u_1|_c = 1$ et donc $|u_2|_c = 0$ ainsi que $|u_3u_4|_c = 1$
   (a) $|u_3|_c = 1$ et donc $|u_4|_c = 0$; on a, par conséquent, que $|w_1|_c = |w_2|_c = 1$, et en utilisant cette règle on en déduit qu'on ne peut pas produire de mot dans L.
   (b) $|u_4|_c = 1$ et donc $|u_3|_c = 0$, de sorte que nous obtenons une contradiction, car dans ce cas, $|w_1|_c = 2$.

3'. Il en va de même si $|u_2|_c = 1$ et que, par conséquent, $|u_1|_c = 0$ et $|u_3u_4|_c = 1$.

En résumé, nous avons des règles de deux types. Les règles du premier type produisent des mots ayant une occurrence de c, ces mots ne peuvent donc pas appartenir à L. Si nous supposons que nous pouvons produire L en utilisant les règles de l'autre type, nous pouvons alors montrer pour chaque règle de cette sorte un exemple de mots tels qu'en leur appliquant cette règle un des mots résultants contiendra deux occurrences de c, de sorte qu'on aura un mot qui ne fait pas partie de $\mathcal{L}$. Ce qui contredit le fait que $\mathcal{L}$ est fermé par rapport à la recombinaison. Par conséquent, nous obtenons une contradiction avec le fait que $\mathcal{L} = L + LcL$ est un langage de recombinaison double. □

Nous ne savons pas si le langage ci-dessus est un langage de recombinaison simple, mais nous pensons que c'est le cas.

Conjecture 3.2.2. Soit $L \subseteq V^*$ un langage rationnel et $c \notin V$. Le langage $L + LcL$ est dans $H_1(FIN, FIN)$, mais pas dans $H_2(FIN, FIN)$.

Une des possibilités de résolution de cette conjecture, c'est-à-dire que $L + LcL \in H_1(FIN, FIN)$, est étroitement lié à la solution proposée pour la conjecture 3.2.1, à savoir que $L + Lc \in H_2(FIN, FIN)$. Nous rappelons que la conjecture 3.2.1 est vraie si nous pouvons trouver $z_i \in V^*$ et $w_i \in L$ tels que pour tout xw_iy dans L les deux conditions suivantes soient satisfaites :

a) $xw_i \in L$,

b) $\forall w \in L : w = w'z_i \Rightarrow wy = w'z_iy \in L$.

Dans notre cas, les règles $z_i\#c\$cw_i\#\varepsilon$ permettent de produire L à partir de LcL. La motivation de cet énoncé est la même que pour la conjecture 3.2.1, mais dans ce cas le problème est encore plus simple car la condition a) est toujours vraie.

Exemple 3.2.4.

$\mathcal{L} = a^* + a^*ba^* \in H_1(FIN, FIN) \setminus H_2(FIN, FIN)$.

En vue du théorème 3.2.7, nous obtenons $\mathcal{L} \notin H_2(FIN, FIN)$. Maintenant, nous utilisons la remarque donnée plus haut et nous posons $w_1 = a$ et $z_1 = \varepsilon$. Il est facile d'observer que la condition b) est satisfaite. La règle $\varepsilon\#b\$ba\#\varepsilon$ permet donc de produire a^* à partir de a^*ba^*. Pour finir la preuve nous remarquons que a^*ba^* est un langage de recombinaison double et, par conséquent, de recombinaison simple, voir théorème 3.2.1.

Exemple 3.2.5.

Si nous considérons le langage L de l'exemple 3.2.2, nous obtenons immédiatement que $L + LcL$ est un langage de recombinaison simple. Le même résultat est vrai pour le langage L de l'exemple 3.2.3.

3.2.3 Exemples de classes de langages rationnels qui ne sont pas des langages de recombinaison simple

Nous présentons ici une classe de langages rationnels qui ne sont pas des langages de recombinaison simple. Cette classe est construite à partir du langage cLc, $L \subseteq V^*$, $c \notin V$.

Théorème 3.2.8. *Soit $L \subseteq V^*$ un langage rationnel et $c \notin V$. Le langage $\mathcal{L}$ défini par $\mathcal{L} = L + cLc$ n'est pas un langage de recombinaison simple.*

Démonstration. Nous allons prouver l'énoncé par contradiction. Supposons qu'il existe un système de recombinaison simple $S = (V, A, R)$ tel que $L(S) = \mathcal{L}$. Nous considérons chaque règle r de R et nous montrons qu'en utilisant cette règle nous obtenons soit un mot ne faisant pas partie de $\mathcal{L}$ parce qu'il contient une seule occurrence de c, soit un mot qui contient deux c, de sorte que nous ne pouvons pas produire de mot de L en utilisant cette règle. Nous remarquons aussi la propriété de fermeture de $\mathcal{L} = \sigma_1(\mathcal{L})$ ou σ est le schéma de recombinaison (V, R).

Considérons donc une règle $r = u_1\#u_2\$u_3\#u_4 \in R$ et soient $x = x_1u_1u_2x_2 \in \mathcal{L}$ et $y = y_1u_3u_4y_2 \in \mathcal{L}$. Nous posons w le résultat de l'application de la règle r sur ces deux mots.

Nous observons que $|u_1u_2|_c \leq 2$ et $|u_3u_4|_c \leq 2$. Il faut donc considérer les cas où chaque $|u_i|_c$, $i = 1 \ldots 4$, varie entre 0 et 2. Nous pouvons écarter les cas où $|u_1|_c > 0$ ou bien le cas où $|u_4|_c > 0$, car dans ces cas nous ne pouvons pas obtenir de mot de L.

À présent, fixons $|u_1|_c = |u_4|_c = 0$ et considérons $0 \leq |u_2|_c, |u_3|_c \leq 2$. Cela nous donne 9 cas. Nous numérotons les cas par deux chiffres ; le premier chiffre indique le nombre de c dans u_2 et le deuxième chiffre indique le nombre de c dans u_3.

Cas 00 : soit $|u_2|_c = 0 = |u_3|_c$. Il existe donc x', y' dans L tels que $(cx'c, y') \vdash_r w$ et $|w|_c = 1$, et donc $w \notin \mathcal{L}$.

Cas 01 : soit $|u_2|_c = 0$ et $|u_3|_c = 1$. Prenons un y qui contienne le site u_3u_4 de la règle r.
Si $u_4y_2 \neq \varepsilon$, il existe x' dans L tel que $(x', y) \vdash_r w$ et $|w|_c = 1$ puisque $|u_4y_2|_c = 1$.
Si $u_4y_2 = \varepsilon$, il existe x' dans L tel que $(cx'c, y) \vdash_r w$ puisque $w = x_1u_1$ et $|w|_c = 1$.
Dans les deux cas $w \notin \mathcal{L}$.

Cas 02 : soit $u_4 = \varepsilon$. Il existe donc x' dans L tel que $(cx'c, y) \vdash_r w$ et $|w|_c = 1$ puisque $w = x_1u_1$, de sorte que $w \notin \mathcal{L}$.

Cas 10 : Ce cas est similaire au cas 01. Nous prenons y' dans L et nous choisissons y' ou $cy'c$ en fonction de x et nous obtenons que $|w|_c = 1$, ce qui entraîne $w \notin \mathcal{L}$.

Cas 11 : ici nous avons 4 sous-cas et $x, y \in \mathcal{L}$:
a) $x_1u_1 = u_4y_2 = \varepsilon$: nous obtenons $w = \varepsilon$.

b) $x_1u_1, u_4y_2 \neq \varepsilon$: comme $|u_2|_c = |u_3|_c = 1$ nous obtenons $|x_1u_1|_c = |x_1|_c = 1$ et $|u_4y_2|_c = |y_2|_c = 1$. Dans ces conditions, $|w|_c = 2$ et w ne peut donc pas être dans L.

c,d) soit $x_1u_1 = \varepsilon$, soit $u_4y_2 = \varepsilon$. Si $x_1u_1 = \varepsilon$, on a $w = u_4y_2$ et $|w|_c = 1$. Si $u_4y_2 = \varepsilon$, on a $w = x_1u_1$ et $|w|_c = 1$. Dans les deux cas, $w \notin \mathcal{L}$.

Cas 12 : soit $u_4 = \varepsilon$. Il existe donc x' dans L tel que $(cx'c, y) \vdash_r w$ et soit $w = \varepsilon$, ce qui se produit si $x_1u_1 = \varepsilon$, soit $|w|_c = 1$.

Cas 20 : soit $u_1 = \varepsilon$. D'une manière similaire au cas 02, nous pouvons trouver y' dans L tel que $(x, cy'c) \vdash_r w$ et $|w|_c = 1$.

Cas 21 : soit $u_1 = \varepsilon$. D'une manière similaire au cas 12, nous obtenons qu'il existe y' dans L tel que $(x, cy'c) \vdash_r w$ et soit $w = \varepsilon$, ce qui se produit quand $u_4y_2 = \varepsilon$, soit $|w|_c = 1$.

Cas 22 : Nous produisons ε.

En résumé, les règles sont de trois types. Les règles du premier type permettent de produire uniquement ε ; les règles du deuxième type permettent d'obtenir des mots w tels que $|w|_c = 2$. Par conséquent, en utilisant ces deux types de règles nous ne pouvons pas obtenir des mots de L. Les règles du troisième type permettent d'obtenir des mots d'un type différent. Mais pour chaque règle, nous pouvons indiquer des exemples de mots de L et cLc tels que le résultat w de l'application de la règle à ces mots contienne une seule occurrence de c, et, de ce fait, $w \notin \mathcal{L}$. C'est une contradiction avec le fait que $\mathcal{L}$ est fermé par rapport à la recombinaison. Par conséquent, nous obtenons une contradiction avec le fait que $\mathcal{L}$ est un langage de recombinaison simple. □

3.3 Conclusions

Nous avons introduit dans ce chapitre l'opération de recombinaison et les systèmes de Head. Nous nous sommes intéressés à la comparaison de la puissance d'expression des systèmes de Head fondés sur la recombinaison simple et des systèmes de Head fondés sur la recombinaison double et nous avons montré que l'ensemble $H_1(FIN, FIN) \setminus H_2(FIN, FIN)$ n'est pas vide. Les langages utilisés pour montrer ce résultat sont construits à partir des langages constants où la constante est définie par une lettre $c \notin V$. Nous remarquons que nous pouvons remplacer cette lettre par des mots qui seront des constantes pour le langage considéré. Même si ces langages peuvent paraître plus intéressants, leur structure est identique à celle des langages présentés plus haut.

Les systèmes de Head introduits dans ce chapitre présentent un grand intérêt, mais leur puissance d'expression n'est pas grande. C'est pourquoi nous allons les enrichir avec des mécanismes de contrôle et de distribution du calcul, ce qui permettra d'atteindre la puissance d'expression d'une machine de Turing. Nous allons complexifier progressivement les systèmes obtenus et le chapitre suivant présente une première extension des systèmes de Head.

Chapitre 4

Systèmes distribués à changement de phase

Dans ce chapitre nous introduisons une extension des systèmes de Head : les systèmes de Head étendus. Nous montrons que ces derniers sont un peu plus puissants que les systèmes de Head en montrant qu'ils peuvent produire la famille des langages rationnels lorsqu'un ensemble fini d'axiomes et de règles est utilisé, ou la famille des langages récursivement énumérables, lorsqu'un ensemble rationnel de règles est utilisé. Nous décrivons aussi la méthode «faire-tourner-et-simuler» utilisée souvent dans le cadre des systèmes de Head et de leurs extensions pour simuler une grammaire arbitraire, ce qui permet donc de montrer leur grande puissance d'expression. Nous introduisons aussi une autre extension des systèmes de Head : les systèmes distribués à changement de phase qui sont fondés sur l'observation biologique que les enzymes, simulés par des règles, ne sont pas tous accessibles dans le même temps, mais à certains moments. Nous montrons que deux jeux de règles, que nous appellerons plus tard *composants*, suffisent pour produire tous les langages récursivement énumérables en simulant une grammaire arbitraire ; la preuve de ce résultat est fondée sur la méthode «faire-tourner-et-simuler». À la fin du chapitre nous montrons quelques résultats obtenus à l'aide du logiciel développé pendant le DEA de l'auteur. Plus précisément, nous avons contrôlé quelques articles publiés sur les systèmes distribués à changement de phase et nous y avons trouvé des erreurs à l'aide de notre logiciel. Nous montrons aussi comment il est possible de corriger ces erreurs dans certains cas.

4.1 Systèmes de Head étendus

4.1.1 Définitions et résultats

Définition 4.1.1. Un *système de Head étendu* est un quadruplet $\gamma = (V, T, A, R)$, où V est un alphabet fini, $T \subseteq V$ est un alphabet terminal, $A \subseteq V^*$ est un ensemble de mots initiaux, appelés axiomes, et R est un ensemble de règles de recombinaison.

Le langage produit par le système de Head étendu γ est

$L(\gamma) = \sigma^*(L) \cap T^*$, où $\sigma = (V, R)$.

Le langage produit par le système de Head étendu γ consiste donc en tous les mots produits par le système de Head $H = (V, A, R)$ qui font partie de l'alphabet terminal T.

Nous notons par $EH(\mathcal{F}_1, \mathcal{F}_2)$ la famille des langages produits par les systèmes de Head étendus dont l'ensemble des axiomes appartient à la famille $\mathcal{F}_1$ et dont l'ensemble des règles appartient à la famille $\mathcal{F}_2$.

Les systèmes de Head étendus sont un peu plus puissants que les systèmes non-étendus.

Théorème 4.1.1. *$REG \subseteq EH(FIN, FIN)$*

Démonstration. Nous présentons ici la preuve donnée dans [44].

Soit $L \subseteq T^*$ un langage rationnel produit par une grammaire rationnelle G, et soit $G = (N, T, S, P)$.

Nous construisons le système de Head étendu suivant :

$$\gamma = (N \cup T \cup \{Z\}, T, A_1 \cup A_2 \cup A_3, R_1 \cup R_2),$$

où

$$\begin{aligned}
A_1 &= \{S\}, \\
A_2 &= \{ZaY \mid X \to aY \in P, X, Y \in N, a \in T\}, \\
A_3 &= \{ZZa \mid X \to a \in P, X \in N, a \in T\}, \\
R_1 &= \left\{ \frac{\varepsilon \mid X}{Z \mid aY} \mid X \to aY \in P, X, Y \in N, a \in T \right\}, \\
R_2 &= \left\{ \frac{\varepsilon \mid X}{ZZ \mid a} \mid X \to a \in P, X \in N, a \in T \right\}.
\end{aligned}$$

Si nous effectuons une recombinaison avec le mot ZxX en utilisant une règle de R_1, le résultat sera de la forme $ZxaY$ et ZX. Il est facile de voir qu'à partir de ces mots nous ne pouvons pas obtenir de mot terminal. En effet, nous ne pouvons pas éliminer le symbole Z si ces mots sont utilisé comme premier terme de la recombinaison. De plus, aucune règle n'est applicable au mot ZxX, où $|x| \neq 1$, si nous essayons de l'utiliser comme deuxième terme de la recombinaison. Par conséquent, la seule possibilité d'obtenir un mot terminal est de partir de S en utilisant les règles de R_1 et de terminer avec une règle de R_2. Ainsi, nous pouvons voir que le premier terme de la recombinaison est celui que nous obtenons par la recombinaison précédente et que le deuxième appartient à A_2 ou à A_3, pour la dernière étape. Ceci correspond à une dérivation dans G. Nous obtenons donc $L(\gamma) = L(G) = L$. □

Il est clair que l'inclusion inverse a lieu, car la famille des langages rationnels est close par rapport à l'intersection.

4.1.2 La méthode «faire-tourner-et-simuler»

Maintenant, il suffit de faire un petit pas en avant dans le cadre de notre modèle pour obtenir une grande puissance d'expression.

Théorème 4.1.2. $RE \subseteq EH(FIN, REG)$

Démonstration. Nous présentons ici la preuve donnée dans [44].

Soit $G = (N, T, S, P)$ une grammaire arbitraire. Posons $U = N \cup T \cup \{B\}$, où B est un nouveau symbole. Considérons le système de Head étendu suivant

$$\gamma = (V, T, A, R),$$

où

$$V = N \cup T \cup \{X, X', B, Y, Z\} \cup \{Y_\alpha \mid \alpha \in U\},$$

$$\begin{aligned} A = {} & \{XBSY,\ ZY, XZ\} \\ & \cup \{ZvY \mid u \to v \in P\} \\ & \cup \{ZY_\alpha,\ X'\alpha Z \mid \alpha \in U\}, \end{aligned}$$

et R contient les groupes de règles suivants :

Simuler :	1.	$Xw\#uY\$Z\#vY$,	pour $u \to v \in P, w \in U^*$,
Faire tourner :	2.	$Xw\#\alpha Y\$Z\#Y_\alpha$,	pour $\alpha \in U, w \in U^*$,
	3.	$X'\alpha\#Z\$X\#wY_\alpha$,	pour $\alpha \in U, w \in U^*$,
	4.	$X'w\#Y_\alpha\$Z\#Y$,	pour $\alpha \in U, w \in U^*$,
	5.	$X\#Z\$X'\#wY$,	pour $w \in U^*$,
Terminer :	6.	$\varepsilon\#ZY\$XB\#wY$,	pour $w \in T^*$,
	7.	$\varepsilon\#Y\$XZ\#\varepsilon$.	

Nous affirmons que $L(\gamma) = L(G)$.

Posons $\sigma = (V, R)$. Nous examinons le fonctionnement de σ, plus exactement les possibilités d'obtention de mot dans T^*.

Aucun mot de A n'est dans T^*. Toutes les règles de R exigent un mot contenant le symbole Z, mais ce symbole n'apparaît pas dans le premier mot produit par la recombinaison. De plus, le deuxième résultat de la recombinaison contient Z et ne peut contenir de site que de la règle 7, mais il est facile d'observer qu'on ne peut pas obtenir un mot terminal dans ce cas. À chaque étape, nous devons donc recombiner un mot de A avec le résultat de la recombinaison de l'étape précédente, excepté à la première étape, quand l'axiome $XBSY$ est utilisé.

Le symbole B marque le début des formes syntaxiques de G simulées par σ.

Les règles du groupe 1 permettent de simuler les règles de P. Les règles des groupes 2 à 5 transfèrent des symboles de la droite du mot considéré à sa gauche. Ce procédé permet la simulation des règles de P à la droite du mot produit par σ. Comme le symbole B qui marque le début du mot est toujours présent, nous pouvons reconstituer le mot à chaque fois. Plus précisément, si le mot considéré dans σ est

de la forme $\beta_1 w_1 B w_2 \beta_2$, où β_1, β_2 sont des parenthèses de type X, X', Y, Y_α, avec $\alpha \in U$, $w_2 w_1$ est une forme syntaxique de G.

Nous partons de $XBSY$, c'est-à-dire de l'axiome de G marqué à gauche par B et entouré par les parenthèses X, Y.

Nous allons voir maintenant comment fonctionnent les règles 2 à 5. Considérons un mot $Xw\alpha Y$ pour $\alpha \in U$, et pour $w \in U^*$. En utilisant une règle du type 2 nous obtenons :

$$(Xw|\alpha Y, Z|Y_\alpha) \vdash (XwY_\alpha, Z\alpha Y).$$

Le symbole Y_α mémorise le fait que α a été effacé de la droite de $w\alpha$. Seule une règle du type 3 peut s'appliquer maintenant à XwY_α :

$$(X'\alpha|Z, X|wY_\alpha) \vdash (X'\alpha wY_\alpha, XZ).$$

Nous remarquons que le symbole α, effacé à l'étape précédente, est placé devant w. Maintenant, il n'existe qu'une possibilité de continuer : c'est d'utiliser une règle du type 4 :

$$(X'\alpha w|Y_\alpha, Z|Y) \vdash (X'\alpha wY, ZY_\alpha).$$

Si nous utilisons maintenant une règle du type 7, alors X' (et B) ne pourront jamais être éliminés, le mot ne pourra donc jamais devenir terminal. Cela nous oblige à utiliser une règle du type 5 :

$$(X|Z, X'|\alpha wY) \vdash (X\alpha wY, X'Z).$$

Nous sommes partis de $Xw\alpha Y$ et nous avons obtenu $X\alpha wY$. Nous pouvons répéter ce processus un nombre de fois arbitraire. Par conséquent nous obtenons toutes les permutations circulaires du mot considéré et rien de plus, car à chaque étape nous avons une occurrence de B.

On voit facilement que chaque règle de P peut être simulée, où que soit son site d'application dans une forme syntaxique de G. Pour cela, nous faisons tourner le mot d'une lettre par les règles 2 à 5 comme décrit plus haut, jusqu'à ce que le membre gauche de la règle à simuler arrive à la fin de la forme syntaxique que l'on fait tourner. Puis nous pouvons appliquer une règle du type 1 qui simule l'application de cette règle de la grammaire.

Par conséquent, pour chaque forme syntaxique w de G, il existe un mot de la forme $XBwY$ produit par σ et, réciproquement, si Xw_1Bw_2Y est produit par σ, w_2w_1 est une forme syntaxique de G.

La seule possibilité d'effacer les symboles ne faisant pas partie de T, est d'utiliser les règles des groupes 6 et 7. Plus précisément, les symboles X et B peuvent être effacés dans les conditions suivantes :

(1) Y est présent : nous ne pouvons donc pas utiliser la règle 7 avant la règle 6 pour effacer Y, car sinon, on ne pourrait plus appliquer de recombinaison au mot résultant qui, de ce fait, ne pourrait être terminal.

(2) Le mot présent est entouré par des parenthèses X et Y et il contient uniquement des lettres terminales

(3) B est juste après X.

Dans ce cas nous pouvons effacer XB, puis Y et le résultat est un mot terminal v. En vue des réflexions présentées plus haut, il est clair que v est dans $L(G)$, donc $L(\gamma) \subseteq L(G)$. Inversement, chaque mot de $L(G)$ peut être produit de cette façon, et donc $L(G) \subseteq L(\gamma)$. Cela donne l'égalité $L(G) = L(\gamma)$, ce qui prouve l'énoncé. □

Nous utilisons dans ce qui suit plusieurs variantes de cette technique appelée «faire-tourner-et-simuler». Cette technique venant de la réécriture a été adaptée par Gh. Păun dans [38] pour le cas des systèmes de Head.

4.2 Systèmes distribués à changement de phase

L'approche suivie ci-dessus utilise des objets infinis, elle est donc purement mathématique. Il est difficile de voir pour l'instant son utilité d'un point de vue biologique. Plus bas, nous présentons un modèle qui permet d'avoir une grande puissance d'expression, tout en gardant le caractère fini du système. Pour cela il suffit d'introduire un peu de distribution dans les calculs, ainsi qu'une procédure de contrôle, ce qui est peut être plus réaliste en l'état actuel de nos connaissances dans le domaine de la biologie.

Définition 4.2.1. Un *système distribué à changement de phase* de degré n est le $n+3$-uplet suivant

$$D = (V, T, A, R_1, R_2, \ldots, R_n),$$

où V est un alphabet, $T \subseteq V$ est l'alphabet terminal, $A \subseteq V^*$ est un ensemble fini d'axiomes et où R_i, $1 \leq i \leq n$, les composants, sont des ensembles finis de règles de recombinaison.

À chaque instant $k = n \cdot j + i$, où $j \geq 0$, $1 \leq i \leq n$, seules les règles du composant R_i sont utilisées pour la recombinaison des mots existants. Plus précisément, nous définissons

$L_1 = A$,
$L_{k+1} = \sigma_i(L_k)$, pour $i \equiv k - 1 \pmod{n} + 1$, $k \geq 1, 1 \leq i \leq n$, $\sigma_i = (V, R_i)$.

En effet, à chaque étape k, les mots présents, L_k, sont recombinés une fois selon les règles du composant R_i, $i \equiv k-1 \pmod{n}+1$ et le résultat de cette recombinaison forme l'ensemble des mots suivants, L_{k+1} ; les mots restants sont éliminés.

Nous disons que le composant R_i d'un système distribué à changement de phase rejette le mot w, s'il n'existe pas de mot complémentaire à w par rapport à une règle de R_i. Dans ce cas, on écrit $w \uparrow_{R_i}$. Nous pouvons omettre R_i si le contexte le permet. En particulier, w est rejeté par R_i si w ne s'accorde avec aucune règle de R_i.

Le langage produit par un système distribué à changement de phase D consiste en tous les mots faisant partie de l'alphabet terminal et qui sont produits à une étape quelconque du calcul.

$$L(D) \stackrel{\text{def}}{=} (\cup_{k\geq 1} L_k) \cap T^*.$$

Nous notons par VDH_n la famille des langages produits par les systèmes distribués à changement de phase de degré au plus n.

Exemple 4.2.1.

Considérons le système distribué à changement de phase suivant :
$D = (V, T, A, R)$, où $V = T = \{a, b, c\}$, $A = \{cab\}$, $R = \left\{ r = \begin{array}{c|c} c & a \\ \hline a & b \end{array} \right\}$.

Nous avons $L_1 = \{cab\}$. Nous pouvons appliquer la règle r à cab et à lui-même : $(c|ab, ca|b) \vdash_r (cb, caab)$. Cela donne $L_2 = \{cb, caab\}$. Nous pouvons maintenant appliquer r à $caab$: $(c|aab, caa|b) \vdash_r (cb, caaaab)$, ce qui nous donne $L_2 = \{cb, caaaab\}$. On voit facilement que $L_k = \{cb, ca^{2^k}b\}$, le langage produit par D est donc le suivant : $L(D) = \{cb, ca^{2^n}b\}$, $n \geq 0$ qui n'est pas un langage algébrique.

4.3 La puissance d'expression des systèmes distribués à changement de phase

Nous voyons que les systèmes distribués à changement de phase sont assez puissants, même avec un seul composant. Dans ce chapitre nous montrons que deux composants suffisent pour obtenir la même puissance d'expression que celle d'une machine de Turing. Nous observons que le même résultat peut être obtenu avec un seul composant et nous présentons cette preuve au chapitre 6, voir théorème 6.3.1.

Théorème 4.3.1. *Soit $G = (N, T, P, S)$ une grammaire arbitraire. Il existe un système distribué à changement de phase de degré 2, $D_G = (V, T, A, R_1, R_2)$ qui simule G et pour lequel $L(G) = L(D_G)$.*

Démonstration.

Définition du système

Posons $D_G = (V, T, A, R_1, R_2)$.

Soient $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(a_n = B)$ et $B \notin N \cup T$.

Dans ce qui suit nous supposons que : $1 \leq \mathbf{i} \leq n$, $1 \leq \mathbf{j} \leq n-1$, $2 \leq \mathbf{k} \leq n$ et $\mathbf{a} \in N \cup T \cup \{B\}$.

L'alphabet V est défini par $V = N \cup T \cup \{B\} \cup \{X, Y, Z, Z', Z'', X_\mathbf{i}, Y_\mathbf{i}, X'_\mathbf{j}, Y'_\mathbf{j}, X''_\mathbf{j}, Y''_\mathbf{j}, X', Y', X'', Y'', C_1, C_2, D_1, D_2\}$

L'alphabet terminal T est celui de la grammaire formelle G.

Les axiomes sont définis par : $A = \{XSBY\} \cup \{ZY_\mathbf{i}, ZY'_\mathbf{j}, ZY''_\mathbf{j}, X'Z, X''Z, ZY, X_\mathbf{i}a_\mathbf{i}Z, X'_\mathbf{j}Z, X''_\mathbf{j}Z, ZY'ZY'', XZ, X_\mathbf{j}Z, C_1Z'D_1, Z''C_2, D_2\} \cup \{ZvY_\mathbf{j}, \exists u \to va_\mathbf{j} \in P\}$.

Les composants du système sont définis de la manière suivante.
Composant R_1 :

$$1.1 : \frac{\varepsilon \mid uY}{Z \mid vY_{\mathbf{j}}}, \quad \exists u \to va_{\mathbf{j}} \in P; \qquad 1.1' : \frac{\varepsilon \mid a_{\mathbf{i}}uY}{Z \mid Y_{\mathbf{i}}}, \quad \exists u \to \varepsilon \in P;$$

$$1.2 : \frac{\varepsilon \mid a_{\mathbf{i}}Y}{Z \mid Y_{\mathbf{i}}}; \qquad 1.3 : \frac{\mathbf{a} \mid Y_{\mathbf{k}}}{Z \mid Y'_{\mathbf{k}-1}}; \qquad 1.4 : \frac{\mathbf{a} \mid Y'_{\mathbf{j}}}{Z \mid Y''_{\mathbf{j}}};$$

$$1.5 : \frac{X_1 \mid \mathbf{a}}{X' \mid Z}; \qquad 1.6 : \frac{X' \mid \mathbf{a}}{X'' \mid Z}; \qquad 1.7 : \frac{\mathbf{a} \mid Y''}{Z \mid Y};$$

$$1.8 : \frac{\mathbf{a} \mid BY''}{Z'' \mid \varepsilon}; \qquad 1.9 : \frac{\mathbf{a} \mid Y''_{\mathbf{j}}}{Z \mid Y_{\mathbf{j}}}; \qquad 1.10 : \frac{C_1 \mid Z'}{\varepsilon \mid D_1};$$

Composant R_2 :

$$2.1 : \frac{X \mid \mathbf{a}}{X_{\mathbf{i}}a_{\mathbf{i}} \mid Z}; \quad 2.2 : \frac{X_{\mathbf{k}} \mid \mathbf{a}}{X'_{\mathbf{k}-1} \mid Z}; \quad 2.3 : \frac{X'_{\mathbf{j}} \mid \mathbf{a}}{X''_{\mathbf{j}} \mid Z}; \quad 2.4 : \frac{\mathbf{a} \mid Y_1}{Z \mid Y'};$$

$$2.5 : \frac{\mathbf{a} \mid Y'}{Z \mid Y''}; \quad 2.6 : \frac{X'' \mid \mathbf{a}}{X \mid Z}; \quad 2.7 : \frac{X'' \mid \mathbf{a}}{\varepsilon \mid Z'}; \quad 2.8 : \frac{X''_{\mathbf{j}} \mid \mathbf{a}}{X_{\mathbf{j}} \mid Z}; \quad 2.9 : \frac{Z'' \mid C_2}{D_2 \mid \varepsilon};$$

Les composants R_1 and R_2 contiennent aussi les règles suivantes :
$\frac{\alpha \mid \varepsilon}{\alpha \mid \varepsilon}$ pour chaque axiome $\alpha \in A$, sauf $XSBY$.

Nous affirmons que $L(D_G) = L(G)$.

Nous allons prouver cette affirmation de la manière suivante. Nous montrerons comment il est possible de simuler les dérivations de la grammaire formelle G et, par conséquent, nous obtiendrons que $L(G) \subseteq L(D_G)$. Dans le même temps, nous montrerons que toutes les autres possibilités de calcul n'aboutissent pas à un mot terminal, ce qui prouvera l'énoncé.

Notre simulation est fondée sur la méthode «faire-tourner-et-simuler», voir section 4.1.2. Nous utilisons une technique similaire à celle qui est utilisée dans [23, 40, 44]. Nous commençons avec le mot Xwa_iY dans le composant R_1. Après l'application de la règle 1.2, le composant R_2 reçoit le mot XwY_i. Dans R_2 la règle 2.1 est appliquée et R_1 reçoit les mots $X_j a_j w Y_i$ $(1 \leq j \leq n)$. À partir de cet instant, le système D_G fait décroître les indices i et j simultanément. Les mots pour lesquels $j \neq i$ sont éliminés et seuls les mots $X_1 a_i w Y_1$ restent. Puis, nous obtenons le mot Xa_iwY, nous avons donc fait tourner le mot Xwa_iY d'une lettre. Si $a_iw = a_iw'B$ et $a_iw' \in T^*$, ce dernier mot faisant partie du résultat est aussi produit.

Schéma du calcul

Le calcul dans D_G suit le schéma présenté à la figure 4.1. Les sommets du schéma représentent une configuration de mots pendant le calcul. Nous énumérons toutes les configurations possibles et nous mettons leur numéro dans le coin supérieur droit.

Le symbole **w** dans les configurations est considéré comme une variable et il prend des valeurs différentes dans des configurations différentes. Par exemple, si dans la configuration 1 le symbole **w** est égal à $w'a_i$, dans la configuration 2 sa valeur peut être w'. Nous allons montrer que le calcul suit le schéma de la figure 4.1, c'est-à-dire que tous les mots produits dans une configuration seront éliminés, excepté les mots appartenant à la configuration suivante.

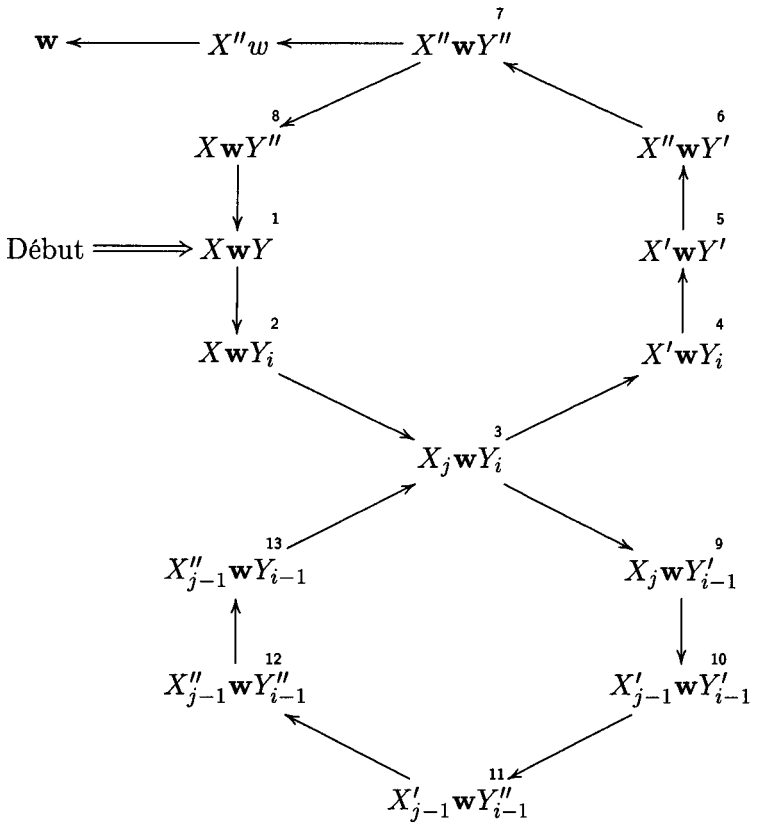


FIG. 4.1 – Le schéma du calcul

Nous remarquons que la règle 1.10 permet de produire le mot Z' à partir des axiomes C_1Z' et D_1. Ce mot apparaîtra donc dans le deuxième composant. D'une manière similaire, la règle 2.9 permet de produire le mot Z'' qui apparaîtra dans le première composant. Cette astuce est utilisée afin d'éviter un calcul erroné et la sous-section 4.5.1 contient plus d'information sur ce sujet.

Rotation

Considérons le mot Xwa_iY ($w \in (N \cup T \cup \{B\})^*$), $1 \leq i \leq n$. Nous sommes dans la configuration 1. Nous allons montrer comment la rotation de wa_i est effectuée.

$(Xw|a_iY, Z|Y_i) \vdash_{1.2} (XwY_i, Za_iY \uparrow)$.

Le mot Za_iY ne peut s'accorder avec aucune règle de R_2, il est donc éliminé.

Maintenant nous sommes dans la configuration 2.

$(X|wY_i, X_k a_k|Z) \vdash_{2.1} (\mathbf{X_k a_k wY_i}, XZ)$, $1 \leq i, k \leq n$.

Le mot XZ est un axiome.

Si $i = 1$, nous pouvons avoir aussi le calcul suivant :

$(Xw|Y_1, Z|Y') \vdash_{2.4} (XwY' \uparrow, ZY_1)$.

Le mot ZY_1 est un axiome et le mot XwY' est éliminé.

Nous sommes dans la configuration 3. Il existe, maintenant, 4 cas possibles :

a) $X_j a_s wY_1$; b) $X_1 a_s wY_i$; c) $X_j a_s wY_i$; d) $X_1 a_s wY_1$; $i, j > 1$, $1 \leq s \leq n$

a) $\mathbf{X_j a_s wY_1}, j > 1$.

Le mot $X_j a_s wY_1$ ne peut s'accorder avec aucune règle de R_1, il est donc éliminé.

b) $\mathbf{X_1 a_s wY_i}, i > 1$.

Ici on a deux possibilités :

(i) $(X_1 a_s w|Y_i, Z|Y'_{i-1}) \vdash_{1.3} (X_1 a_s wY'_{i-1} \uparrow, ZY_i)$.

Le mot ZY_i est un axiome. Le mot $X_1 a_s wY'_{i-1}$ ne peut s'accorder avec aucune règle de R_2, il est donc éliminé.

(ii) $(X_1|a_s wY_i, X'|Z) \vdash_{1.5} (X' a_s wY_i \uparrow, X_1 Z)$.

Le mot $X_1 Z$ est un axiome. Le mot $X' a_s wY_i$ ne peut s'accorder avec aucune règle de R_2, il est donc éliminé.

Dans ces deux cas, le calcul n'aboutit donc pas à de nouveaux mots.

c) $\mathbf{X_j a_s wY_i}, i, j > 1$.

$(X_j a_s w|Y_i, Z|Y'_{i-1}) \vdash_{1.3} (X_j a_s wY'_{i-1}, ZY_i)$.

Le mot ZY_i est un axiome.

Nous sommes dans la configuration 9.

$(X_j|a_s wY'_{i-1}, X'_{j-1}|Z) \vdash_{2.2} (X'_{j-1} a_s wY'_{i-1}, X_j Z)$.

Le mot $X_j Z$ est un axiome.

Nous sommes dans la configuration 10.

$(X'_{j-1} a_s w|Y'_{i-1}, Z|Y''_{i-1}) \vdash_{1.4} (X'_{j-1} a_s wY''_{i-1}, ZY'_{i-1})$.

Le mot ZY'_{i-1} est un axiome.

Nous sommes dans la configuration 11.

$(X'_{j-1}|a_s wY''_{i-1}, X''_{j-1}|Z) \vdash_{2.3} (X''_{j-1} a_s wY''_{i-1}, X'_{j-1} Z)$.

Le mot $X'_{j-1} Z$ est un axiome.

Nous sommes dans la configuration 12.

$(X''_{j-1} a_s w|Y''_{i-1}, Z|Y_{i-1}) \vdash_{1.9} (X''_{j-1} a_s wY_{i-1}, ZY''_{i-1})$.

Le mot ZY''_{i-1} est un axiome.

Nous sommes dans la configuration 13.

Maintenant il existe deux possibilités de poursuivre le calcul.

(i) $(X''_{j-1} a_s w|Y_1, Z|Y') \vdash_{2.4} (X''_{j-1} a_s wY' \uparrow, ZY_1)$.

Le mot ZY_1 est un axiome et le mot $X''_{j-1} a_s wY'$ est éliminé.

(ii) $(X''_{j-1}|a_s wY_{i-1}, X_{j-1}|Z) \vdash_{2.8} (\mathbf{X_{j-1} a_s wY_{i-1}}, X''_{j-1} Z)$.

Le mot $X''_{j-1} Z$ est un axiome. Nous observons que nous sommes arrivé à nouveau dans la configuration 1. Il est facile de voir que les indices de X et Y ont été diminués simultanément. Nous pouvons continuer jusqu'à ce que l'un des indices devienne 1, nous serons donc dans l'un des cas (a), (b) ou (d).

d) $\mathbf{X_1 a_s w Y_1}$.

$(X_1|a_s w Y_1, X'|Z) \vdash_{1.5} (X' a_s w Y_1, X_1 Z)$.

Le mot $X_1 Z$ est un axiome.

Nous sommes dans la configuration 4.

$(X' a_s w|Y_1, Z|Y') \vdash_{2.4} (X' a_s w Y', Z Y_1)$.

Le mot $Z Y_1$ est un axiome.

Nous sommes dans la configuration 5.

$(X'|a_s w Y', X''|Z) \vdash_{1.6} (X'' a_s w Y', X' Z)$.

Le mot $X' Z$ est un axiome.

Nous sommes dans la configuration 6.

Nous avons maintenant trois possibilités :

(i) $(X''|a_s w Y', X|Z) \vdash_{2.6} (X a_s w Y' \uparrow, X'' Z)$.

Le mot $X'' Z$ est une axiome. Le mot $X a_s w Y'$ ne pouvant s'accorder avec aucune règle de R_1, il est éliminé.

(ii) $(X''|a_s w Y', |Z') \vdash_{2.7} (a_s w Y' \uparrow, X'' Z' \uparrow)$.

Les mots $a_s w Y'$ et $X'' Z'$ ne pouvant s'accorder avec aucune règle de R_1, ils sont éliminés.

(iii) $(X'' a_s w|Y', Z|Y'') \vdash_{2.5} (X'' a_s w Y'', Z Y')$.

Le mot $Z Y'$ est un axiome.

Nous sommes dans la configuration 7.

(*) $(X'' a_s w|Y'', Z|Y) \vdash_{1.7} (X'' a_s w Y, Z Y'')$.

Le mot $Z Y''$ est un axiome.

Nous sommes dans la configuration 8.

(**) $(X''|a_s w Y, X|Z) \vdash_{2.6} (\mathbf{X a_s w Y}, X'' Z)$.

Le mot $X'' Z$ est un axiome.

Nous sommes arrivé dans la configuration 1, nous avons donc fait tourner le symbole a_s. Il est possible d'appliquer les règles 1.8 et 2.7 dans les cas marqués par (*) et (**), mais ces applications seront discutées plus tard.

Simulation des productions de la grammaire

S'il existe une production $u \to v a_i$ (ou $u \to \varepsilon$) et un mot $X w' u Y$ ($X w'' a_i u Y$), nous pouvons appliquer la règle 1.1 (1.1').

$(X w'|u Y, Z|v Y_i) \vdash_{1.1} (X w' v Y_i, Z u Y \uparrow)$.

Le mot $Z u Y$ ne pouvant s'accorder avec aucune règle de R_2 est éliminé.

$(X w''|a_i u Y, Z|Y_i) \vdash_{1.1'} (X w'' Y_i, Z a_i u Y \uparrow)$.

Le mot $Z a_i u Y$ ne pouvant s'accorder avec aucune règle de R_2 est éliminé.

Puis, le système fonctionne comme dans le cas de la rotation. Par conséquent, nous simulons l'application de la règle $u \to v a_i$ ($u \to \varepsilon$) de la grammaire G.

Obtention du résultat

Si nous avons le symbole B à la fin du mot dans le cas (*), nous pouvons appliquer la règle 1.8 :

$(X''a_sw'|BY'', Z''|) \vdash_{1.8} (X''a_sw', Z''BY'' \uparrow)$.
Le mot $Z''BY''$ est éliminé.

$(X''|a_sw', |Z') \vdash_{2.7} (\mathbf{a_sw'} \uparrow, X''Z' \uparrow)$.
Les mots $X''Z'$ et a_sw' sont éliminés. Si le mot $a_sw' \in T^*$, a_sw' fait partie du résultat du calcul.

Il était possible d'appliquer la règle 2.6 :
$(X''|a_sw', X|Z) \vdash_{2.6} (Xa_sw' \uparrow, X''Z)$.
Le mot $X''Z$ est un axiome et Xa_sw' est éliminé.

Nous observons que par définition nous avons à chaque étape un nombre illimité de copies de chaque mot. Dans le cas ci-dessus, les deux applications se font donc en parallèle et une de ces applications produit le résultat du calcul, tandis que l'autre n'aboutit pas à de nouveaux mots.

Considérons maintenant le cas (**). Il est possible d'appliquer la règle 2.7 :
$(X''|a_swY, |Z') \vdash_{2.7} (a_swY, X''Z' \uparrow)$.
Le mot $X''Z'$ est éliminé.

$(a_sw'|a_tY, Z|Y_t) \vdash_{1.1,\ 1.1'\ ou\ 1.2} (a_sw'Y_t, Za_tY \uparrow),\ 1 \leq t \leq n$.
Le mot Za_tY est éliminé.

Si $t \neq 1$, le mot $a_sw'Y_t$ est éliminé.
Si $t = 1$,
$(a_sw'|Y_1, Z|Y') \vdash_{2.4} (a_sw'Y' \uparrow, ZY_1)$.
Le mot ZY_1 est un axiome et le mot $a_sw'Y'$ est éliminé.
Ce calcul n'aboutit donc à aucun mot nouveau.

Autres calculs avec les axiomes

Nous pouvons avoir les calculs suivants entre les axiomes, mais ils n'aboutissent pas à des mots nouveaux.

$(Zv|Y_i, Z|Y'_{i-1}) \vdash_{1.3} (ZvY'_{i-1} \uparrow, ZY_i),\ 2 \leq i \leq n$.

$(X_1|a_1Z, X'|Z) \vdash_{1.5} (X'a_1Z \uparrow, X_1Z)$.

$(X_k|a_kZ, X'_{k-1}|Z) \vdash_{2.2} (X'_{k-1}a_kZ \uparrow, X_kZ),\ 2 \leq k \leq n$.

$(Zv|Y_1, Z|Y') \vdash_{2.4} (ZvY' \uparrow, ZY_1)$.
Comme nous avons examiné tous les cas, notre analyse est maintenant complète.

Remarques finales

Nous observons qu'il était possible d'utiliser le mot C_1Z' à la place de Z' et $Z''C_2$ à la place de Z'' dans certains endroits du calcul ci-dessus, mais ces applications, introduisant C_1 et C_2 dans le mot considéré, ne changent pas le fait du rejet du mot par le composant suivant.

Il est facile de voir qu'en suivant le schéma du calcul nous obtenons tous les mots de $L(G)$ et, comme nous avons considéré tous les cas possibles, il est clair que le système ne produit pas d'autres mots. □

4.4 Un système distribué à changement de phase « petit » qui est universel

Nous allons montrer que les systèmes distribués à changement de phase avec un seul composant sont indécidables. Pour cela nous montrons qu'avec un seul composant il est possible de simuler un système de Post simplifié. De même, comme les systèmes de Post simplifiés ont été utilisées pour obtenir des machines de Turing de taille petite, voir [46], le système obtenu a une description qui est très compacte.

Nous définissons aussi un *entrée* pour un système distribué à changement de phase. Une entrée pour le système D est un mot w sur l'alphabet de D. Le calcul de D sur l'entrée w est effectué en ajoutant w aux axiomes et en laissant le calcul de D se dérouler comme d'habitude.

Théorème 4.4.1. *Soit $T = (2, V, P)$ un système de Post simplifié et $w \in V^*$. Il existe un système distribué à changement de phase D de degré 1, $D = (V', V, A, R_1)$, qui en lui donnant comme entrée le mot $LXwY$ simule T sur l'entrée w, c'est-à-dire que :*

1. *pour chaque mot w sur lequel T s'arrête produisant le résultat w', le système D produit un résultat unique w'.*
2. *pour chaque mot w, sur lequel T ne s'arrête pas, le système D calcule infiniment sans produire de résultat.*

Démonstration.

Soient $V = \{a_1, \ldots, a_{n+1}\}$ et $P = \{P_1, \ldots, P_n\}$.

Dans ce qui suit nous considérons que $1 \leq i \leq n$ et $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \{a_1, \ldots, a_n\}$.

Nous définissons D de la manière suivante.

L'alphabet V' est défini par : $V' = V \cup \{X, Y, L, X', Y', R, Z\}$.

Les axiomes sont définis par : $A = \{LXwY, ZP_iY, X'E, RY'\}$.

Les règles de R_1 sont définies de la manière suivante :

Règles principales :

$$1 : \frac{LXa_i\mathbf{b} \mid \mathbf{c}}{X' \mid E} \; ; \quad 2 : \frac{\mathbf{c}Y \mid \varepsilon}{L \mid Xa_i\mathbf{b}E} \; ; \quad 3 : \frac{\mathbf{a} \mid YXa_i}{Z \mid P_iY} \; ; \quad 4 : \frac{X' \mid \mathbf{a}}{LX \mid E} \; ;$$

Règles secondaires :

$$5 : \frac{LX \mid a_i\mathbf{b}E}{LXa_i \mid \mathbf{b}E} \; ; \quad 6 : \frac{LX \mid \mathbf{b}E}{LX\mathbf{b} \mid E} \; ; \quad 7 : \frac{ZP_i \mid Y}{R \mid Y'} \; ; \quad 8 : \frac{ZP_i \mid Y'}{R \mid Y} \; ;$$

Règles pour le résultat :

$$9 : \frac{LXa_{n+1}\mathbf{b} \mid \mathbf{c}}{\varepsilon \mid X'E} \; ; \qquad 10 : \frac{\mathbf{c} \mid Y}{LXa_{n+1}\mathbf{b}X'E \mid \varepsilon} \; ;$$

Nous montrons maintenant comment on simule le calcul de T. Nous avons dans le système un mot de la forme $LXwY$ qui correspond au mot w sur lequel on travaille dans T. Nous détachons d'abord les deux premiers symboles de w et nous marquons par X' le début du mot, ce qui nous donne deux molécules : Xa_ia_jE et $X'w'Y$, où $w = a_ia_jw'$. Puis nous collons la partie détachée à la fin du mot en obtenant $X'w'YXa_ia_jE$, ce qui permet de la remplacer par la bonne production : $X'w'P_iY$. Finalement, on enlève le prime de X en obtenant : $LXw'P_iY$ et on peut recommencer le processus. Nous nous arrêtons lorsque le premier symbole est a_{n+1}. Dans ce cas, nous enlevons les parenthèses LX et Y et nous obtenons le mot-résultat.

Nous détaillons par la suite ce que nous venons de dire, en considérant toutes les évolutions possibles.

Étape 1.

$$\frac{LXa_ib \mid wY}{X' \mid E} \vdash_1 \frac{X'wY}{LXa_ibE},$$

$$\frac{ZP_i \mid Y}{R \mid Y'} \vdash_7 \frac{ZP_iY'}{RY}.$$

Étape 2.

$$\frac{X'wY \mid \varepsilon}{L \mid Xa_ibE} \vdash_2 \frac{X'wYXa_ibE}{L\uparrow},$$

$$\frac{LX \mid a_ibE}{LXa_i \mid bE} \vdash_5 \frac{LXbE}{LXa_ia_ibE\uparrow},$$

$$\frac{ZP_i \mid Y'}{R \mid Y} \vdash_8 \frac{ZP_iY}{RY'}.$$

Étape 3.

$$\frac{X'w \mid YXa_ibE}{Z \mid P_iY} \vdash_3 \frac{X'wP_iY}{ZYXa_ibE\uparrow},$$

$$\frac{LX \mid bE}{LXb \mid E} \vdash_6 \frac{LXE}{LXbbE\uparrow},$$

$$\frac{ZP_i \mid Y}{R \mid Y'} \vdash_7 \frac{ZP_iY'}{RY}.$$

Étape 4.

$$\frac{X' \mid wP_iY}{LX \mid E} \vdash_4 \frac{LXwP_iY}{X'E},$$

$$\frac{ZP_i \mid Y'}{R \mid Y} \vdash_8 \frac{ZP_iY}{RY'}.$$

Nous observons que nous avons simulé une étape de calcul dans T et que toutes les molécules non-désirables ont été éliminées. Nous pouvons continuer ce processus jusqu'à ce que le premier symbole de w devienne a_{n+1}. Comme nous avons besoin de quatre étapes pour simuler une étape de T, la situation ci-dessus arrive lorsque on est à l'étape $4k+1$, $k \geq 0$. Dans ce cas nous pouvons enlever les parenthèses LX et Y et obtenir le mot-résultat :

Étape 4k+1.

$$\frac{LXa_{n+1}b \mid wY}{\varepsilon \mid X'E} \vdash_9 \frac{wY}{LXa_{n+1}bX'E}.$$

Étape 4k+2.

$$\frac{w \mid Y}{LXa_{n+1}bX'E \mid \varepsilon} \vdash_{10} \frac{w\uparrow}{LXa_{n+1}bX'EY\uparrow}.$$

Comme nous n'aurons plus de mot de la forme $LXwY$, le calcul suivant consistera en application des règles 7 et 8 sans produire un mot terminal.

□

4.5 Correction des systèmes existants

Comme nous l'avons dit au Chapitre 2, nous avons utilisé un logiciel de simulation des systèmes distribués à changement de phase, développé pendant le DEA de l'auteur. Nous avons contrôlé plusieurs articles existants sur les systèmes distribués à changement de phase et nous avons trouvé que certains d'entre eux contient des erreurs. Dans cette section nous présentons les systèmes correspondants, nous montrons les erreurs trouvées, ainsi que les corrections nécessaires pour que les systèmes retrouvent un fonctionnement correct. Nous avons trouvé que certaines erreurs sont communes à plusieurs auteurs, nous espérons donc que notre correction permettra de les éviter dans le futur.

Chacun des systèmes considérés simule une grammaire arbitraire $G=(V,T,S,P)$.

4.5.1 Correction du système TVDH3

Nous commençons par le système distribué à changement de phase présenté dans [23]. La figure 4.2 contient la description de ce système, appelé TVDH3.

Le fonctionnement du système TVDH3 n'est pas correct. Les auteurs n'ont pas observé la situation suivante : l'application de la règle 2.5 produit le mot $X'Z'$ qui ne fait pas partie des axiomes et qui n'est pas éliminé du système grâce à la règle $Z'\#\varepsilon\$Z'\#\varepsilon$. Puis, à l'aide de la règle 2.4, on obtient XZ'. Ce mot utilisé avec la règle 2.1 produit les mots X_ia_iZ' et X_ma_iZ', où $m \neq i$. Finalement, la règle 2.5 permet d'introduire ces préfixes dans n'importe quel mot. Plus précisément :

$(X'|wY', |Z') \vdash_{2.5} (X'Z', wY')$.

$X'Z'$ est présent à chaque étape grâce à la règle $Z'\#\varepsilon\$Z'\#\varepsilon$.

$(X'|Z', X|Z) \vdash_{2.4} (XZ', X'Z)$.

Maintenant, la règle 2.1 s'applique à XZ', qui est toujours présent, et à un de ses mots complémentaires X_ia_iZ :

$(X|Z', X_ia_i|Z) \vdash_{2.1} (XZ, X_ia_iZ')$.

Les mots X_ia_iZ' sont toujours présents. À partir d'eux on peut obtenir $X'_{i-1}a_iZ'$:

$(X_i|a_iZ', X'_{i-1}|Z) \vdash_{1.3} (X_iZ, X'_{i-1}a_iZ')$.

On continue et on obtient d'une manière similaire $X''_{i-1}a_iZ'$, $X_{i-1}a_iZ'$, et finalement $X'a_iZ'$ et Xa_iZ'.

Maintenant, soit $X'wY'$ une évolution correcte. La règle 2.5 du deuxième composant permet alors d'insérer a_i devant w :

$(X'|wY', Xa_i|Z') \vdash_{2.5} (X'Z', Xa_iwY')$,

ce qui n'est pas une évolution correcte.

Considérons le système TVDH3 $= (V, T, A, R_1, R_2, R_3)$.

Soient $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(a_n = B)$ et $B \notin N \cup T$.

Soient i, j et k avec $1 \leq i \leq n$, $1 \leq j \leq n-1$, $2 \leq k \leq n$.

L'alphabet V est défini par $V = N \cup T \cup \{B\} \cup \{X, Y, Z, X_i, Y_i, X'_j, Y'_j, X''_j, Y''_j, X', Y', Z', Z''\}$.

Les axiomes A sont donnés par $A = \{XSBY\} \cup \{ZvY_i, \exists u : u \to va_i \in P,\ i \in \{1, \ldots, n-1\}\} \cup \{ZY_i, X'_jZ, X'Z,\ X_ia_iZ,\ ZY''_j,\ X_jZ,\ XZ,\ ZY'_j,\ ZY',\ X''_jZ,\ ZY, Z', Z''\}$.

Composant R_1 :

$1.1 : \dfrac{\varepsilon \mid uY}{Z \mid vY_j}$, $\exists u \to va_j \in P$; $1.2 : \dfrac{\varepsilon \mid a_iY}{Z \mid Y_i}$; $1.3 : \dfrac{X_k \mid \varepsilon}{X'_{k-1} \mid Z}$;

$1.4 : \dfrac{X_1 \mid \varepsilon}{X' \mid Z}$; $1.5 : \dfrac{\varepsilon \mid Y''_i}{Z \mid Y_i}$; $1.6 : \dfrac{\varepsilon \mid a_iuY}{Z \mid Y_i}$, $\exists u \to \varepsilon \in P$;

Composant R_2 :

$2.1 : \dfrac{X \mid \varepsilon}{X_ia_i \mid Z}$; $2.2 : \dfrac{\varepsilon \mid Y'_j}{Z \mid Y''_j}$; $2.3 : \dfrac{X''_j \mid \varepsilon}{X_j \mid Z}$;

$2.4 : \dfrac{X' \mid \varepsilon}{X \mid Z}$; $2.5 : \dfrac{X' \mid \varepsilon}{\varepsilon \mid Z'}$;

Composant R_3 :

$3.1 : \dfrac{\varepsilon \mid Y_k}{Z \mid Y'_{k-1}}$; $3.2 : \dfrac{X'_j \mid \varepsilon}{X''_j \mid Z}$; $3.3 : \dfrac{\varepsilon \mid Y_1}{Z \mid Y'}$;

$3.4 : \dfrac{\varepsilon \mid Y'}{Z \mid Y}$; $3.5 : \dfrac{\varepsilon \mid BY'}{Z'' \mid \varepsilon}$;

Note.

Les composants R_1, R_2 et R_3 contiennent aussi les règles suivantes :

$\dfrac{\alpha \mid \varepsilon}{\alpha \mid \varepsilon}$ pour chaque axiome $\alpha \in A$, sauf $XSBY$.

FIG. 4.2 – Le système TVDH3

On voit bien que le problème est causé par le mot XZ'. Pour le résoudre il suffit de remplacer la règle 2.4 par l'ensemble de règles suivant :

$\dfrac{X' \mid a_i}{X \mid Z}$, $1 \leq i \leq n$.

Dans ce cas le mot XZ' ne peut pas être produit et le système reprend un fonctionnement correct.

Nous présentons ici encore une solution au problème ci-dessus qui sera utilisée dans le futur. Cette solution concerne les mots qui se terminent par Z' : wZ'. Nous observons, que le dysfonctionnement du système TVDH3 est du au fait que ces mots, une fois produits, persistent tout le temps grâce à la règle $Z'\#\varepsilon\$Z'\#\varepsilon$. Nous pouvons remplacer cette règle par l'ensemble de règles suivant :

$$\frac{C \mid Z'}{D \mid \varepsilon} \in R_1 \text{ et } \left\{ \frac{CZ' \mid \varepsilon}{CZ' \mid \varepsilon}, \frac{D \mid \varepsilon}{D \mid \varepsilon} \right\} \in R_1 \cap R_2 \cap R_3.$$

Il faut ajouter aussi CZ' et D aux axiomes. Dans ce cas, seul CZ' est persistant et tous les autres mots se terminant en Z' sont éliminés, ce qui résoud notre problème.

Nous observons que dans le système présenté dans le théorème 4.3.1 cette deuxième approche a été utilisée pour éviter le même problème que celui posé dans la section présente.

4.5.2 Correction du système TVDH4

Nous continuons avec le système présenté dans [35] que nous appelons TVDH4. Sa définition est donnée à la figure 4.3.

Nous supposons que la grammaire G est dans la forme normale de Kuroda. Nous posons $N \cup T \cup \{F\} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ $(\alpha_n = F)$ et $F \notin \{N \cup T\}$. Soient i, j et s avec $1 \leq i \leq l$, $m+1 \leq j \leq l$ et $1 \leq s \leq m$.
Soient $u_k \to v_k \in P$, $n+1 \leq k \leq m$ les règles hors contexte de P, et soient $u_k \to v_k \in P$, $m+1 \leq k \leq l$ les autres règles de P. Posons $u_k = v_k = \alpha_k$ pour $1 \leq k \leq n$.

Considérons le système TVDH4 $= (V, T, A, R_1, R_2, R_3, R_4)$.

où

$V = N \cup T \cup \{X, Y, Z, Z', Z_0, Z_0', F, X_0, Y_0, X_i, Y_i, Y_j', Z_i'\}$,

$A = \{XSFY, XZ', ZY, Z_0, Z_0', X_0Z', ZY_0, X_iZ', ZY_i, X_iv_iZ_i', ZY_j'\}$,

$$R_1 = Q \cup \left\{1.1 : \frac{\varepsilon \mid Y}{\varepsilon \mid Y},\ 1.2 : \frac{X_0 \mid \varepsilon}{X_0 \mid \varepsilon},\ 1.3 : \frac{\varepsilon \mid Y_i}{\varepsilon \mid Y_i},\ 1.4 : \frac{\varepsilon \mid Y_j'}{\varepsilon \mid Y_j'}\right\},$$

$$R_2 = Q \cup \left\{2.1 : \frac{X \mid \varepsilon}{X \mid \varepsilon},\ 2.2 : \frac{\varepsilon \mid Y_0}{\varepsilon \mid Y_0},\ 2.3 : \frac{X_i \mid \varepsilon}{X_i \mid \varepsilon}\right\},$$

$$R_3 = Q \cup \left\{3.1 : \frac{\varepsilon \mid Y_0}{Z \mid Y},\ 3.2 : \frac{\varepsilon \mid FY_0}{Z_0 \mid \varepsilon},\ 3.3 : \frac{\varepsilon \mid u_sY}{Z \mid Y_s},\ 3.4 : \frac{\varepsilon \mid Y_i}{Z \mid Y_{i-1}}\right\}$$
$$\cup \left\{3.5 : \frac{\varepsilon \mid EY}{Z \mid Y_j'},\ 3.6 : \frac{\varepsilon \mid DY_j'}{Z \mid Y_{j-1}} \text{ où } u_j = DE,\ D, E \in N\right\},$$

$$R_4 = Q \cup \left\{4.1 : \frac{X_0 \mid \varepsilon}{X \mid Z'},\ 4.2 : \frac{X_0 \mid \varepsilon}{\varepsilon \mid Z_0'},\ 4.3 : \frac{X \mid \varepsilon}{\varepsilon \mid Z_i'},\ 4.4 : \frac{X_i \mid \varepsilon}{X_{i-1} \mid Z'}\right\},$$

et

$$Q = \left\{z_1 : \frac{Z \mid \varepsilon}{Z \mid \varepsilon},\ z_2 : \frac{\varepsilon \mid Z'}{\varepsilon \mid Z'},\ z_3 : \frac{Z_0 \mid \varepsilon}{Z_0 \mid \varepsilon},\ z_4 : \frac{\varepsilon \mid Z_0'}{\varepsilon \mid Z_0'},\ z_5 : \frac{\varepsilon \mid Z_i'}{\varepsilon \mid Z_i'}\right\}.$$

FIG. 4.3 – Le système TVDH4

Le fonctionnement du système TVDH4 n'est pas correct. Il y a trois erreurs :

1) Une erreur du même type que dans l'exemple précédent : la règle 4.2 permet d'obtenir le mot X_0Z_0' qui n'est pas éliminé. Puis, comme plus haut nous pouvons obtenir XZ_0', $X_iv_iZ_0'$, Xv_iZ_0', etc. Considérons maintenant une évolution correcte X_0wY. En utilisant la règle 4.2 on insère v_i devant w :

$(X_0|wY, Xv_i|Z_0') \vdash_{4.2} (X_0Z_0', Xv_iwY)$,

ce qui n'est pas un fonctionnement correct.

Pour retrouver un fonctionnement correct, la règle 4.2 doit être remplacée par l'ensemble de règles $X_0\#\alpha_i\$X\#Z'$, $1 \leq i \leq n$.

2) Une application répétée de la règle 4.4 produit des mots $X_kv_jZ_j'$, $k < j$ qui deviennent persistants. Maintenant la règle 4.3 permet d'insérer X_ka_j, $j \neq k$ devant

un mot, ce qui conduit à une rotation incorrecte.

Pour résoudre ce problème il suffit de remplacer les règles 4.3 par $X\#\varepsilon\$X_jv_j\#Z'_j$.

3) Ce cas présente un problème similaire avec celui du cas (1), mais à l'autre extrémité du mot. La règle 3.2 permet de produire le mot Z_0FY_0 qui est persistant. Une fois produit, ce mot introduit des erreurs dans le calcul :

$(X_1w|FY_0, Z_0|FY_0) \vdash_{3.2} (X_1wFY_0, Z_0FY_0)$.

$(X_1|wFY_0, X_0|Z') \vdash_{4.4} (X_0wFY_0, X_1Z')$.

Puis, X_0wFY_0 arrive inchangé dans le troisième composant, c'est-à-dire que le mot incorrect X_1wFY_0 devient X_0wFY_0 qui est un mot correct.

Nous n'avons pas trouvé de solution à ce problème, une réécriture complète du système est donc nécessaire.

4.5.3 Correction du système TVDH7

Nous continuons avec le système présenté dans [44] que nous appelons TVDH7. Sa définition est donnée à la figure 4.4.

Soit $N \cup T = \{\alpha_1, \ldots, \alpha_{n-1}\}, n \geq 3$, et $P = \{u_i \to v_i \mid 1 \leq i \leq m\}$. Soit $\alpha_n = B$ un nouveau symbole. Soient i et j avec $1 \leq i \leq m$ et $1 \leq j \leq n$.
Posons TVDH7 $= (V, T, A, R_1, \ldots, R_7)$, où

$$\begin{aligned} V &= N \cup T \cup \{X, Y, Y', Z, B, Y_0, X_0, Y'_0, Y_j, Y'_j, X_j\}, \\ A &= \{XBSY, ZY, ZY', ZZ, Zv_iY, ZY_j, ZY'_j, X_j\alpha_j Z, X_jZ\}, \end{aligned}$$

$R_1 = \{1.1: \varepsilon\#u_iY\$Z\#v_iY,\ 1.2: \varepsilon\#Y\$Z\#Y,\ 1.3: \varepsilon\#Y_j\$Z\#Y_j\}$,
$R_2 = \{2.1: \varepsilon\#\alpha_jY\$Z\#Y_j,\ 2.2: \varepsilon\#Y\$Z\#Y',\ 2.3: \varepsilon\#Y_j\$Z\#Y'_j\}$,
$R_3 = \{3.1: X\#\varepsilon\$X_j\alpha_j\#Z,\ 3.2: \varepsilon\#Y'\$Z\#Y,\ 3.3: \varepsilon\#Y'_j\$Z\#Y_j\}$,
$R_4 = \{4.1: \varepsilon\#Y_j\$Z\#Y_{j-1},\ 4.2: \varepsilon\#Y\$Z\#Y\}$,
$R_5 = \{5.1: X_j\#\varepsilon\$X_{j-1}\#Z,\ 5.2: \varepsilon\#Y\$Z\#Y\}$,
$R_6 = \{6.1: \varepsilon\#Y_0\$Z\#Y,\ 6.2: \varepsilon\#Y_0\$ZZ\#\varepsilon,\ 6.3: \varepsilon\#Y\$Z\#Y',\ 6.4: \varepsilon\#Y_j\$Z\#Y'_j\}$,
$R_7 = \{7.1: X_0\#\varepsilon\$X\#Z,\ 7.2: X_0B\#\$\#ZZ,\ 7.3: \varepsilon\#Y'\$Z\#Y,\ 7.4: \varepsilon\#Y'_j\$Z\#Y_j\}$.

Tous les composants contiennent la règle $z.1 : Z\#\varepsilon\$Z\#\varepsilon$.

FIG. 4.4 – Le système TVDH7

Le fonctionnement du système TVDH7 n'est pas correct. Il présente les erreurs suivantes :

1) Il faut ajouter ZY_0, ZY'_0 et X_0Z dans les axiomes.

2) Le problème de la règle $z.1$. L'utilisation de cette règle permet d'obtenir des mots incorrects :

$(X_j\alpha_jZ|, Z|Z) \vdash_{z.1} (X_j\alpha_jZZ, Z)$,

$(X_j\alpha_jZZ|, Z|Y_k) \vdash_{z.1} (X_j\alpha_jZZY_k, Z)$.

Maintenant nous pouvons obtenir tous les mots $X_i\alpha_jZZY_k$ possibles.

Finalement, on a l'application suivante conduisant à une erreur :
$(X_0B|wY, X\alpha_i|ZZY_k) \vdash_{7.2} (X\alpha_iwY, X_0BZZY_k)$.

Une erreur similaire est causée par la règle 6.2, mais à l'autre extrémité du mot.

Une solution pour cette erreur est de remplacer ZZ par un nouveau symbole Z', mais cela crée un nouveau problème, similaire au problème de TVDH3. Dans ce cas on produit X_0BZ' par la règle 7.2 qui est transformé en $X_i\alpha_iBZ'$ qui, étant utilisé dans la règle 7.2 avec X_0BwY_i, conduit à un mot erroné. On peut résoudre ce deuxième problème par la deuxième méthode présentée pour TVDH3 en plaçant les règles correspondantes dans le composant 6.

4.6 Conclusions

Les systèmes distribués à changement de phase introduits dans ce chapitre possèdent beaucoup de propriétés intéressantes. Grâce à la stratégie d'élimination ils ont un contrôle très puissant qui permet de les manipuler facilement et qui les rend très simples. C'est pourquoi ces systèmes peuvent jouer un rôle important, car leur structure simple donne la possibilité de les simuler par d'autres types de systèmes fondés sur la recombinaison. Nous verrons aux chapitres 6 et 9 des exemples de systèmes qui simulent des systèmes distribués à changement de phase.

Pour le contrôle des systèmes obtenus nous avons utilisé le logiciel `TVDHsim` qui a été développé pendant le DEA de l'auteur. Ce logiciel permet d'effectuer une simulation d'un système distribué à changement de phase étape par étape. L'utilisation de ce logiciel a permis de découvrir les erreurs dans les systèmes présentés à la section 4.5.

Dans le chapitre suivant nous considérons des systèmes qui ressemblent beaucoup aux systèmes dont nous avons parlé dans ce chapitre et nous pourrons réutiliser certaines idées dans le cadre de ces nouveaux systèmes.

Chapitre 5

Systèmes distribués à changement de phase étendus

Les systèmes distribués à changement de phase présentés dans le chapitre précédent sont fondés sur l'idée suivante : les mots de l'ensemble courant sont recombinés une fois et seul le résultat de cette recombinaison forme l'ensemble des mots suivants. Or, à l'instant, cette recombinaison unique est presque impossible à réaliser dans la pratique. À l'heure actuelle, il est plus naturel, d'un point de vue biologique, de supposer qu'à chaque étape nous pouvons effectuer un nombre illimité de recombinaisons de la même manière que ce qui a été fait dans le cas des systèmes de Head, voir la section 3.1. Ceci nous conduit aux systèmes distribués à changement de phase étendus, introduits par M. Margenstern et Yu. Rogozhin dans [23], où cette considération a été prise en compte. Dans ce chapitre nous donnons une définition formelle de ces systèmes, ainsi qu'une description de leur puissance d'expression.

5.1 Définition formelle

Soit $\sigma = (V, R)$ une schéma de recombinaison. Nous définissons l'opération $\tilde{\sigma}$, *cf.* [23] :

$\tilde{\sigma}(L) = \tilde{\sigma}(L' \cup L'') \stackrel{\text{def}}{=} \sigma^*(L')$, où

$L' = \{w_1 \in L | \exists w_2 \in L : \exists w, w' \in V^* : \exists r \in R : (w_1, w_2) \vdash_r (w, w')$ ou $(w_2, w_1) \vdash_r (w, w')\}$, $L'' = L \setminus L'$.

Plus précisément, pour obtenir $\tilde{\sigma}(L)$ nous prenons les mots de L qui sont complémentaires par rapport à une règle de R (c'est-à-dire L') et nous appliquons σ^* à cet ensemble. Comme résultat nous obtenons toutes les recombinaisons itératives de ces mots, ainsi que les mots initiaux de L'.

Définition 5.1.1. Un *système distribué à changement de phase étendu* de degré n est le $n + 3$-uplet suivant

$$E = (V, T, A, R_1, R_2, \ldots, R_n),$$

où V est un alphabet, $T \subseteq V$ est l'alphabet terminal, $A \subseteq V^*$ est un ensemble fini d'axiomes et où R_i, $1 \leq i \leq n$, les composants, sont des ensembles finis de règles de recombinaison.

À chaque instant $k = n \cdot j + i$, où $j \geq 0$, $1 \leq i \leq n$, seules les règles du composant R_i sont utilisées pour la recombinaison des mots existants. Plus précisément, nous définissons

$L_1 = A$,
$L_{k+1} = \tilde{\sigma}_i(L_k)$, pour $i \equiv k-1 \pmod{n} + 1$, $k \geq 1, 1 \leq i \leq n$, $\sigma_i = (V, R_i)$.

En termes usuels, ceci signifie qu'à chaque étape k le composant R_i d'un système distribué à changement de phase étendu pour lequel $i \equiv k-1 \pmod{n}+1$ applique d'abord un filtre sur l'ensemble des mots courants, c'est-à-dire qu'il élimine les mots ne possédant pas de mot complémentaire par rapport à une des règles de R_i ; puis le système fonctionne comme un système de Head ordinaire.

Nous disons que le composant R_i d'un système distribué à changement de phase étendu rejette le mot w, s'il n'existe pas de mot complémentaire à w par rapport à une règle de R_i. Dans ce cas, on écrit $w \uparrow_{R_i}$. Nous pouvons omettre R_i si le contexte le permet. En particulier, w est rejeté par R_i si w ne s'accorde avec aucune règle de R_i.

Le langage produit par un système distribué à changement de phase étendu E consiste en tous les mots faisant partie de l'alphabet terminal et qui sont produits à une étape quelconque du calcul.

$$L(E) \stackrel{\text{def}}{=} (\cup_{k \geq 1} L_k) \cap T^*.$$

Nous notons par $EVDH_n$ la famille des langages produits par les systèmes distribués à changement de phase étendus de degré au plus n.

Exemple 5.1.1.

Reprenons l'exemple 4.2.1, mais cette fois dans le cadre des systèmes distribués à changement de phase étendus.

$E = (V, T, A, R)$, où $V = T = \{a, b, c\}$, $A = \{cab\}$, $R = \left\{ r = \begin{array}{c|c} c & a \\ \hline a & b \end{array} \right\}$.

Nous avons $L_1 = \{cab\}$. Nous pouvons appliquer la règle r à cab et à lui-même : $(c|ab, ca|b) \vdash_r (cb, caab)$. Cela donne $L_2 = \{cb, cab, caab\}$. Nous pouvons maintenant appliquer r à $caab$: $(c|aab, caa|b) \vdash_r (cb, caaaab)$, ainsi que à cab et à $caab$: $(c|ab, caa|b) \vdash_r (cb, caaab)$, ce qui donne $L_2 = \{cb, cab, caab, caaab, caaaab\}$. On voit facilement que $L_k = \{cb, cab, \ldots, ca^k b\}$, et que le langage produit par E est donc le suivant : $L(D) = ca^n b$, $n \geq 0$, qui est un langage rationnel.

5.2 La puissance d'expression des systèmes distribués à changement de phase étendus

Dans cette section on va s'occuper de la puissance d'expression des systèmes introduits ci-dessus. Maintenant, un seul composant ne suffit plus pour produire

tous les langages récursivement énumérables. Plus exactement, nous avons le résultat suivant :

Théorème 5.2.1. $EVDH_1 = REG$.

Démonstration. Soit $E = (V, T, A, R)$ un système distribué à changement de phase étendu. Selon la définition, on a $L_1 = A$. Soient A' les mots de A qui possèdent un mot complémentaire par rapport à une règle de R. Dans ce cas, $L_2 = \sigma^*(A')$. Soient L_2' les mots de L_2 qui possèdent un mot complémentaire par rapport à une règle de R. Alors, $L_3 = \sigma^*(L_2')$. Il est facile de voir que $L_2 = L_3$. En effet, L_2 est la plus petite clôture de A' par rapport à la recombinaison. Comme $A' \subseteq L_2' \subseteq L_2 = \sigma^*(A')$, on obtient que $\sigma^*(L_2') = L_2$. D'une manière similaire on obtient que $L_k = L_2$, $k > 2$. Le langage produit par E est donc $L(E) = \sigma^*(A') \cap T^*$. Comme $\sigma^*(A')$ est un langage rationnel, voir théorème 3.1.3, et que la famille des langages rationnels est close par rapport à l'intersection, on obtient $L(E) \subseteq REG$. L'inclusion inverse résulte du théorème 4.1.1. □

Nous observons que les systèmes distribués à changement de phase étendus avec un composant ressemblent beaucoup aux systèmes de Head étendus. L'exemple suivant montre la différence entre les deux modèles qui réside dans l'application de la règle de filtrage.

Exemple 5.2.1.

Considérons le système distribué à changement de phase étendu suivant $E = (V, T, A, R)$, où $V = T = \{c, a, b\}$, $A = \{cab, caab\}$ et $R = \left\{ \frac{c \mid aa}{aa \mid b}, \frac{c \mid ab}{aaa \mid b} \right\}$. Considérons aussi le système de Head étendu $EH = (V, T, A, R)$. Il est facile de voir que $L(EH) = ca^nb$, $n \geq 0, n \neq 3$. Pour le système E, le mot cab est éliminé tout de suite, car il ne peut pas participer à une recombinaison. Le langage produit par ce système est donc $L(E) = \sigma^*(\{caab\}) = ca^{2n}b$, $n \geq 0$.

Maintenant, nous montrons que les systèmes distribués à changement de phase étendus ont la même puissance d'expression que les machines de Turing.

Théorème 5.2.2. *Soit $G = (N, T, P, S)$ une grammaire arbitraire. Il existe un système distribué à changement de phase étendu de degré 3, $E_G = (V, T, A, R_1, R_2, R_3)$ qui simule G et pour lequel $L(G) = L(E_G)$.*

Démonstration.

Définition du système

Posons $E_G = (V, T, A, R_1, R_2, R_3)$.

Soit $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(B = a_n)$ et $B \notin N \cup T$.

Dans ce qui suit nous supposons que $1 \leq i \leq n$, $1 \leq j \leq n-1$, $2 \leq l \leq n$, $1 \leq k \leq 5, \mathbf{a} \in N \cup T \cup \{B\}$.

L'alphabet V est défini par $V = N \cup T \cup \{B\} \cup \{X, Y, X_i, Y_i, X_i', Y_i', X_j'', Y_j'', X', Y', X'', Y'', X''', Y''', Y^{IV}, Z, Z_E, C, C_1, C_2, D, D_1, D_2, Z_E^k\}$.

L'alphabet terminal T est celui de la grammaire formelle G.

Les axiomes sont définis par : $A = \{XSBY, X_i'Z_E^3, Z_E^4Y_i', ZY_i, X''Z, ZY^{IV}, X_ia_iZ, ZY_j'', X'Z, ZY', X_jZ, ZY, ZY''', XZ, X_j''Z, ZY'', X'''Z, ZY, CZ_E^k, CZ_E, Z_E^kD, Z_ED, C_2Z'', D_2, Z'C_1, D_1\} \cup \{ZvY : \exists u \to v \in P\}$.

Les composants du système sont définis de la manière suivante.

Composant R_1 :

$1.1 : \frac{\varepsilon \mid uY}{Z \mid vY}$, $\exists u \to v \in P$; $1.2 : \frac{\varepsilon \mid a_iY}{Z \mid Y_i}$; $1.3 : \frac{X_i \mid \mathbf{a}}{X_i' \mid Z_E}$;

$1.4 : \frac{\mathbf{a} \mid Y_j''}{Z \mid Y_j}$; $1.5 : \frac{X' \mid \mathbf{a}}{X'' \mid Z}$; $1.6 : \frac{\mathbf{a} \mid Y'''}{Z \mid Y^{IV}}$;

$1.7 : \frac{X_i' \mid Z_E}{C \mid Z_E^1}$; $1.8 : \frac{X_i' \mid Z_E^3}{C \mid Z_E^4}$; $1.9 : \frac{Z_E^1 \mid Y_i'}{Z_E^2 \mid D}$;

$1.10 : \frac{Z_E^4 \mid Y_i'}{Z_E^5 \mid D}$; $1.11 : \frac{Z' \mid C_1}{D_1 \mid \varepsilon}$;

Composant R_2 :

$2.1 : \frac{X \mid \mathbf{a}}{X_ia_i \mid Z}$; $2.2 : \frac{\mathbf{a} \mid Y_l'}{Z \mid Y_{l-1}''}$; $2.3 : \frac{X_1' \mid \mathbf{a}}{X' \mid Z}$; $2.4 : \frac{\mathbf{a} \mid Y_1'}{Z \mid Y'}$;

$2.5 : \frac{X_j'' \mid \mathbf{a}}{X_j \mid Z}$; $2.6 : \frac{\mathbf{a} \mid Y''}{Z \mid Y'''}$; $2.7 : \frac{\mathbf{a} \mid BY''}{Z' \mid \varepsilon}$; $2.8 : \frac{X''' \mid \mathbf{a}}{X \mid Z}$;

$2.9 : \frac{X_i' \mid Z_E^1}{C \mid Z_E^2}$; $2.10 : \frac{X_i' \mid Z_E^4}{C \mid Z_E^5}$;

$2.11 : \frac{Z_E^2 \mid Y_i'}{Z_E^3 \mid D}$; $2.12 : \frac{Z_E^5 \mid Y_i'}{Z_E \mid D}$; $2.13 : \frac{C_2 \mid Z''}{\varepsilon \mid D_2}$;

Composant R_3 :

$3.1 : \frac{\mathbf{a} \mid Y_i}{Z_E \mid Y_i'}$; $3.2 : \frac{X_l' \mid \mathbf{a}}{X_{l-1}'' \mid Z}$; $3.3 : \frac{\mathbf{a} \mid Y'}{Z \mid Y''}$; $3.4 : \frac{X'' \mid \mathbf{a}}{X''' \mid Z}$;

$3.5 : \frac{X'' \mid \mathbf{a}}{\varepsilon \mid Z''}$; $3.6 : \frac{\mathbf{a} \mid Y^{IV}}{Z \mid Y}$;

$3.7 : \frac{X_i' \mid Z_E^2}{C \mid Z_E^3}$; $3.8 : \frac{X_i' \mid Z_E^5}{C \mid Z_E}$; $3.9 : \frac{Z_E \mid Y_i'}{Z_E^1 \mid D}$; $3.10 : \frac{Z_E^3 \mid Y_i'}{Z_E^4 \mid D}$;

Les composants R_1, R_2 et R_3 contiennent aussi les règles suivantes :

$\frac{\alpha \mid \varepsilon}{\alpha \mid \varepsilon}$ pour chaque axiome $\alpha \in A$, sauf $XSBY$, $X_iZ_E^3$ et $Z_E^4Y_i$.

Nous affirmons que $L(E_G) = L(G)$.

Nous allons prouver cette affirmation de la manière suivante. Nous montrerons comment il est possible de simuler les dérivations de la grammaire formelle G et,

par conséquent, nous obtiendrons que $L(G) \subseteq L(E_G)$. Dans le même temps, nous montrerons que toutes les autres possibilités de calcul n'aboutissent pas à un mot terminal, ce qui prouvera l'énoncé.

Notations

Nous allons utiliser la notation suivante :

$$\frac{w_1 \mid w_2}{w_1' \mid w_2'} \vdash_r \frac{w_1 w_2'^*}{w_1' w_2}, \quad w_1 w_2$$

où $*$ indique une occurrence possible de $\uparrow$. Dans la partie gauche, l'application de la règle r sur $w_1 w_2$ et $w_1' w_2'$ est indiquée, tandis que la partie droite contient les mots résultants $w_1 w_2'$ et $w_1' w_2$. Nous allons aussi utiliser la convention suivante : la partie supérieure des deux cotés contiendra les mots dont nous sommes intéressés, tandis que la partie inférieure contiendra soit un axiome soit un mot contenant Z et qui n'altère pas le calcul ultérieur. Le symbole $\uparrow$ indique que le mot considéré est rejeté par le composant suivant. Le terme optionnel dans la partie droite de la formule, $w_1 w_2$, est omis si ce mot, qui en principe fait partie de l'ensemble des mots suivants, est rejeté par le composant suivant. Si ce n'est pas le cas, il est alors écrit.

Notre simulation est fondée sur la méthode «faire-tourner-et-simuler», voir aussi la section 4.1.2. Nous utilisons une technique similaire à celle qui est utilisée dans [23, 40, 44], voir aussi le théorème 4.3.1. Nous commençons avec le mot Xwa_iY dans le composant R_1. Après l'application de la règle 1.2, le composant R_2 reçoit XwY_i. Dans R_2, la règle 2.1 est appliquée et R_3 reçoit les mots $X_j a_j w Y_i$ $(1 \leq j \leq n)$. À partir de cet instant, le système E_G fait décroître les indices i et j simultanément. Les mots pour lesquels $j \neq i$ sont éliminés et seuls les mots $X_1 a_i w Y_1$ subsistent. Puis, nous obtenons le mot Xa_iwY, nous avons donc fait tourner le mot Xwa_iY d'une lettre. Si $a_i w = a_i w' B$ et $a_i w' \in T^*$, ce dernier mot faisant partie du résultat est aussi produit.

Le système est construit de telle façon que les mots $X_i' Z_E$ apparaissent dans le premier composant seulement pendant une étape paire du calcul, c'est-à-dire que si nous avons ces mots dans le premier composant, la prochaine fois que nous serons dans ce composant, en fait dans 3 étapes, ces mots $(X_i' Z_E)$ n'existeront plus. C'est une propriété importante du système et elle permet d'effectuer une simulation correcte.

Cette propriété est réalisée de la manière suivante. Nous avons dans le système un mot $X_i' Z_E^p$ et à chaque étape, nous incrémentons p modulo 5 en utilisant les règles 1.7, 1.8, 2.9, 2.10, 3.7, et 3.8. Lorsque $p = 0$, nous obtenons $X_i' Z_E$. Nous commençons avec $X_i' Z_E^3$, ce qui permet d'avoir $X_i' Z_E$ dans le premier composant uniquement pendant les étapes paires du calcul.

Une condition analogue est observée pour les mots $Z_E Y_i'$. Ils apparaissent dans le troisième composant seulement pendant une étape impaire du calcul.

Le mot Z' est produit dans le premier composant et il apparaît seulement dans le deuxième composant ; le mot Z'' est produit dans le deuxième composant et il apparaît seulement dans le troisième composant.

Les motivations pour ces dernières affirmations sont les mêmes que pour la méthode des molécules assistantes et elles peuvent être trouvées au chapitre 6 où une approche plus méthodique est développée.

Le schéma du calcul

Le calcul dans E_G suit le schéma présenté à la figure 5.1. Les sommets du schéma représentent une configuration de mots pendant le calcul. Nous énumérons toutes les configurations possibles et nous mettons leur numéro dans le coin supérieur droit. Le symbole **w** dans les configurations est considéré comme une variable et il prend des valeurs différentes dans les différentes configurations. Par exemple, si dans la configuration 14 le symbole **w** est égal à $w'a_i$, dans la configuration 15 sa valeur peut être w'. Nous allons montrer que le calcul suit le schéma de la figure 5.1, c'est-à-dire que tous les mots produits dans une configuration seront éliminés, excepté les mots appartenant à la configuration suivante.

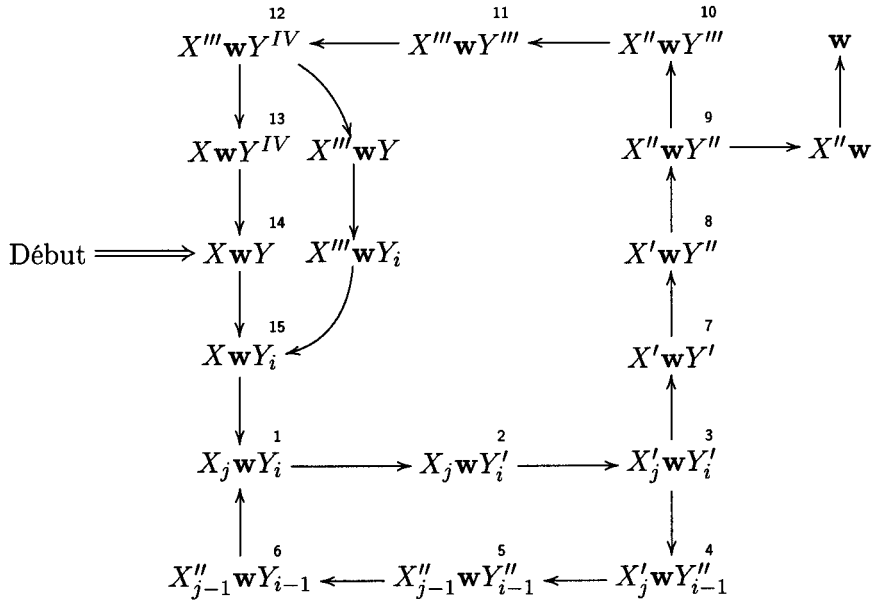


FIG. 5.1 – Le schéma du calcul

Comme la longueur des deux cycles est un nombre pair, la parité de l'étape de calcul est la même pour tous les mots dans une configuration particulière. De plus, il est facile de vérifier que les mots d'une configuration arrivent toujours dans le composant ayant le même numéro. Nous pouvons donc dire que chaque configuration a un numéro de composant et une parité d'étape de calcul associés.

On voit que la décrémentation simultanée des deux indices est effectuée par le cycle inférieur, tandis que le cycle supérieur effectue la rotation des lettres.

Comme nous avons un système distribué à changement de phase étendu, il existe deux types de mots qui passent à l'étape suivante : les mots produits et les mots qui les ont produits, car ces derniers font partie de σ^*. Nous montrons que généralement

ils seront éliminés pendant l'étape suivante, ainsi que les mots produits par ceux-ci pendant l'étape suivante. Nous discutons aussi en détail les autres cas pour montrer que nous effectuons une simulation correcte.

Nous montrons maintenant quelques étapes de calcul, puis nous discuterons chaque configuration en détail.

Nous commençons avec le mot $XwY = Xw'a_iY$, pour un i tel que $1 \leq i \leq n$.

$$\frac{Xw' \mid a_iY}{Z \mid Y_i} \vdash_{1.2} \frac{Xw'Y_i}{Za_iY}, \quad Xw'a_iY\,.$$

Les mots $Xw'a_iY$ et $Xw'Y_i$ passent dans le composant suivant.

Considérons l'évolution de $Xw'a_iY$:

$$\frac{X \mid w'a_iY}{X_ka_k \mid Z} \vdash_{2.1} \frac{X_ka_kw'a_iY\uparrow}{XZ}, \quad k \in \{1,\ldots,n\}.$$

On voit qu'on ne produit pas de mot nouveau. Pour $Xw'Y_i$ nous avons :

$$\frac{X \mid w'Y_i}{X_ka_k \mid Z} \vdash_{2.1} \frac{X_ka_kw'Y_i}{XZ}, \quad Xw'Y_i\,, \quad k \in \{1,\ldots,n\}.$$

Les mots $Xw'Y_i$ et $X_ka_kw'Y_i$ passent dans le composant suivant.

Nous avons une étape impaire. Les mots Z_EY_i' existent donc et nous pouvons appliquer la règle suivante :

$$\frac{Xw' \mid Y_i}{Z_E \mid Y_i'} \vdash_{3.1} \frac{Xw'Y_i'\uparrow}{Z_EY_i}\,.$$

Le mot $Xw'Y_i$ est éliminé. Pour $X_ka_kw'Y_i$ nous obtenons :

$$\frac{X_ka_kw' \mid Y_i}{Z_E \mid Y_i'} \vdash_{3.1} \frac{X_ka_kw'Y_i'}{Z_EY_i}\,.$$

Et les mots $X_ka_kw'Y_i$ et $X_ka_kw'Y_i'$ passent dans le composant suivant.

Et on pourrait continuer de la même façon.

Maintenant, nous discutons précisément les configurations en les regroupant par similitude de comportement ultérieur des calculs, ce que nous présentons ci-après sous sept rubriques intitulées **Groupe** i, pour i de 1 à 7.

Groupe 1 Les configurations $4, 5, 6, 7, 11, 13, 15$ qui ont un comportement simple.

Nous allons discuter en détail la configuration 5.

$$\frac{X''_{j-1}w \mid Y''_{i-1}}{Z \mid Y_{i-1}} \vdash_{1.4} \frac{X''_{j-1}wY_{i-1}}{ZY''_{i-1}}, \quad X''_{j-1}wY''_{i-1}\,, \quad 2 \leq i,j \leq n.$$

Le mot $X''_{j-1}wY_{i-1}$ est dans la configuration 6.

Le mot $X''_{j-1}wY''_{i-1}$ est éliminé pendant l'étape suivante :

$$\frac{X''_{j-1} \mid wY''_{i-1}}{X_{j-1} \mid Z} \vdash_{2.5} \frac{X_{j-1}wY''_{i-1}\uparrow}{X''_{j-1}Z}\,.$$

Un calcul semblable à celui qui est donné ci-dessus et qui nous permet d'avancer dans le schéma du calcul, se produit pour chaque configuration, excepté pour la configuration 3. Dans ce qui suit, nous examinons les autres possibilités de calcul existant pour les autres configurations.

Groupe 2 La configuration 14, liée à l'application de la règle $u \to v$ de la grammaire.

Outre les calculs similaires à ceux du groupe 1, nous avons l'application de la règle 1.1 pour simuler l'application de la règle $u \to v$ de G.

$$\frac{Xw \mid uY}{Z \mid vY} \vdash_{1.1} \frac{XwvY}{ZuY}, \quad XwuY\,.$$

Le mot $XwvY$ peut être utilisé ensuite pour une recombinaison dans le même composant pour un calcul similaire à celui du groupe 1.

Groupe 3 Les configurations 8 et 10 où l'on a les mots Z' ou Z'' complémentaires.

Outre les calculs similaires à ceux du groupe 1 nous avons l'application de la règle 2.7, ou 3.5 pour la configuration 10, mais cela ne conduit pas à de nouveaux mots. Ci-dessous, nous donnons toutes les étapes de ce calcul pour la configuration 10.

$$\frac{X'' \mid wY'''}{\varepsilon \mid Z''} \vdash_{3.5} \frac{wY'''}{X''Z''}, \quad X''wY'''\,,$$

$$\frac{X''w \mid Y'''}{Z \mid Y^{IV}} \vdash_{1.6} \frac{X''wY^{IV}\uparrow}{ZY'''},$$

$$\frac{w \mid Y'''}{Z \mid Y^{IV}} \vdash_{1.6} \frac{wY^{IV}\uparrow}{ZY'''}\,.$$

Groupe 4 La configuration 9 où il est possible de déboucher sur un résultat.

Outre les calculs similaires à ceux des groupes 1 et 3, nous pouvons appliquer les règles 2.7 et 3.5 consécutivement. Dans ce cas nous pouvons ajouter w au résultat si $w \in T^*$.

$$\frac{X''w'' \mid BY''}{Z' \mid \varepsilon} \vdash_{2.7} \frac{X''w''}{Z'BY''}, \quad X''w''BY''\,,$$

$$\frac{X'' \mid w}{\varepsilon \mid Z''} \vdash_{3.5} \frac{\mathbf{w}\uparrow}{X''Z''}\,.$$

Groupe 5 Les configurations 1 et 2.

Outre les calculs similaires à ceux du groupe 1 nous pouvons avoir le calcul suivant, dans le cas de la configuration 1 :

$$\frac{X_1w \mid Y_i}{Z_E \mid Y_i'} \vdash_{3.1} \frac{X_1wY_i'}{Z_EY_i}, \quad X_1wY_i\,, \quad 1 \le i \le n.$$

Le mot X_1wY_i' est dans la configuration 2.

$$\frac{X_1 \mid wY_i}{X_1' \mid Z_E} \vdash_{1.3} \frac{X_1'wY_i}{X_1Z_E}\,.$$

C'est un des cas où un mot générateur produit un mot qui n'est pas éliminé immédiatement.

$$\frac{X_1' \mid wY_i}{X' \mid Z} \vdash_{2.3} \frac{X'wY_i\uparrow}{X_1'Z}\,.$$

Les mots X'_1wY_i et $X'wY_i$ sont rejetés par le composant suivant, car, l'étape suivante étant une étape paire et les mots $Z_EY'_i$ n'existant pas, nous ne pouvons pas appliquer la règle 3.1.

Pour la configuration 2 nous avons le calcul suivant :

$$\frac{X_k \mid wY'_1}{X'_k \mid Z_E} \vdash_{1.3} \frac{X'_kwY'_1}{X_kZ_E}, \quad X_kwY'_1, \quad 1 \leq k \leq n.$$

Le mot $X'_kwY'_1$ est dans la configuration 3.

$$\frac{X_kw \mid Y'_1}{Z \mid Y'} \vdash_{2.4} \frac{X_kwY'}{ZY'_1},$$

$$\frac{X_kw \mid Y'}{Z \mid Y''} \vdash_{3.3} \frac{X_kwY'' \uparrow}{ZY'}.$$

Les mots X_kwY' et X_kwY'' sont rejetés par le composant suivant, car, l'étape suivante étant une étape impaire et les mots X'_iZ_E n'existant pas, nous ne pouvons pas appliquer la règle 1.3.

Groupe 6 La configuration 3 où on contrôle si on est arrivé à la fin de la rotation. Il y a 4 cas selon les valeurs de i et j :

a) $X'_jwY'_1$, b) $X'_1wY'_i$, c) $X'_1wY'_1$, d) $X'_jwY'_i$, $2 \leq i, j \leq n$

a) Le cas $\mathbf{X'_jwY'_1}$.

$$\frac{X'_jw \mid Y'_1}{Z \mid Y'} \vdash_{2.4} \frac{X'_jwY'}{ZY'_1}, \quad X'_jwY'_1,$$

$$\frac{X'_j \mid wY'_1}{X''_{j-1} \mid Z} \vdash_{3.2} \frac{X''_{j-1}wY'_1 \uparrow}{X'_jZ}.$$

Il y a deux règles qui peuvent s'appliquer à X'_jwY' : 3.2 et 3.3. Nous obtenons :

$$\frac{X'_jw \mid Y'}{Z \mid Y''} \vdash_{3.3} \frac{X'_jwY'' \uparrow}{ZY''},$$

$$\frac{X'_j \mid wY'}{X''_{j-1} \mid Z} \vdash_{3.2} \frac{X''_{j-1}wY' \uparrow}{X'_jZ}.$$

Nous pouvons appliquer encore une fois les règles 3.3 et 3.2 aux mots ci-dessus et nous obtenons :

$$\frac{X''_{j-1}w \mid Y'}{Z \mid Y''} \vdash_{3.3} \frac{X''_{j-1}wY'' \uparrow}{ZY'}.$$

Ce calcul ne produit donc pas de nouveaux mots.

b) Le cas $\mathbf{X'_1wY'_i}$.

Il y a deux règles qui peuvent s'appliquer à $X'_1wY'_i$: 2.2 et 2.3. Nous obtenons :

$$\frac{X'_1w \mid Y'_i}{Z \mid Y''_{i-1}} \vdash_{2.2} \frac{X'_1wY''_{i-1} \uparrow}{ZY'_i},$$

$$\frac{X'_1 \mid wY'_i}{X' \mid Z} \vdash_{2.3} \frac{X'wY'_i \uparrow}{X'_1Z}.$$

Nous pouvons appliquer encore une fois les règles 2.3 et 2.2 aux mots ci-dessus, ce qui nous donne :

$$\frac{X'w \mid Y'_i}{Z \mid Y''_{i-1}} \vdash_{2.2} \frac{X'wY''_{i-1}\uparrow}{ZY'_i}.$$

Ce calcul ne produit donc pas de nouveaux mots.

c) Le cas $\mathbf{X'_1wY'_1}$.

Il y a deux règles qui peuvent s'appliquer à $X'_1wY'_1$: 2.4 et 2.3. Nous obtenons :

$$\frac{X'_1w \mid Y'_1}{Z \mid Y'} \vdash_{2.4} \frac{X'_1wY'}{ZY'_1},$$

$$\frac{X'_1 \mid wY'_1}{X' \mid Z} \vdash_{2.3} \frac{X'wY'_1\uparrow}{X'_1Z}.$$

Nous pouvons appliquer encore une fois les règles 2.3 et 2.4 aux mots ci-dessus et nous obtenons :

$$\frac{X'_1 \mid wY'}{X' \mid Z} \vdash_{2.3} \frac{\mathbf{X'wY'}}{X'_1Z}, \quad X'_1wY'.$$

Le mot $X'wY'$ est dans la configuration 7.

$$\frac{X'_1w \mid Y'}{Z \mid Y''} \vdash_{3.3} \frac{X'_1wY''\uparrow}{ZY'}.$$

d) Le cas $\mathbf{X'_jwY'_i}$, $\mathbf{i,j > 1}$.

$$\frac{X'_kw \mid Y'_i}{Z \mid Y''_{i-1}} \vdash_{2.2} \frac{\mathbf{X'_kwY''_{i-1}}}{ZY'_i}, \quad X'_kwY'_i.$$

Le mot $X'_kwY''_{i-1}$ est dans la configuration 4.

$$\frac{X'_k \mid wY'_i}{X''_{k-1} \mid Z} \vdash_{3.2} \frac{X''_{k-1}wY'_i\uparrow}{X'_kZ}.$$

Groupe 7 La configuration 12 qui permet une introduction d'une branche parallèle.

$$\frac{X''' \mid wY^{IV}}{X \mid Z} \vdash_{2.8} \frac{XwY^{IV}}{X'''Z}, \quad X'''wY^{IV}.$$

Le mot XwY^{IV} est dans la configuration 13.

$$\frac{X'''w \mid Y^{IV}}{Z \mid Y} \vdash_{3.6} \frac{X'''wY}{ZY^{IV}},$$

$$\frac{X'''w' \mid a_iY}{Z \mid Y_i} \vdash_{1.2} \frac{X'''w'Y_i}{Za_iY}, \quad X'''w'a_iY, \quad 1 \leq i \leq n,$$

$$\frac{X''' \mid wY}{X \mid Z} \vdash_{2.8} \frac{XwY}{X'''Z},$$

$$\frac{X''' \mid w'Y_i}{X \mid Z} \vdash_{2.8} \frac{Xw'Y_i}{X'''Z}, \quad X'''w'Y_i.$$

Les mots obtenus XwY et $Xw'Y_i$ sont les mêmes que ceux obtenus en suivant le schéma du calcul et ils existent donc pendant cette étape.

$$\frac{X'''w' \mid Y_i}{Z_E \mid Y'_i} \vdash_{3.1} \frac{X'''w'Y'_i\uparrow}{Z_EY_i}.$$

Remarques finales

Nous observons qu'il était possible d'utiliser le mot C_1Z' à la place de Z' et $Z''C_2$ à la place de Z'' dans certains endroits du calcul ci-dessus, mais ces applications, introduisant C_1 et C_2 dans le mot considéré, ne changent pas le rejet du mot par le composant suivant.

Il est facile de voir qu'en suivant le schéma du calcul nous obtenons tous les mots de $L(G)$ et, comme nous avons considéré tous les cas possibles, il est clair que le système ne produit pas d'autres mots.

□

Nous observons que le même résultat peut être obtenu avec deux composants et nous présentons cette preuve au chapitre 6, voir théorème 6.2.1.

5.3 Conclusions

Malgré la complexité des systèmes distribués à changement de phase étendus nous avons réussi à diriger le calcul et à produire tous les langages récursivement énumérables avec trois composants seulement. La démonstration repose sur les mêmes principes que celle du chapitre précédent, mais elle est beaucoup plus compliquée. Il est donc très difficile d'avancer en suivant la même approche. Le chapitre suivant présente une approche différente qui permet de diminuer le nombre des composants jusqu'à deux tout en restant simple.

Chapitre 6

La méthode des molécules assistantes

La technique de preuve utilisée dans les deux chapitres précédents a atteint ses limites et ne peut plus être utilisée pour diminuer le nombre de composants des systèmes considérés. Dans ce chapitre, nous analysons cette technique et nous montrerons une nouvelle méthode qui nous permettra de réduire le nombre de composants des systèmes considérés auparavant. Cette nouvelle technique, la méthode des molécules assistantes, est fondée sur une réorganisation du flux du calcul, ainsi que sur une bonne synchronisation entre différentes parties de celui-ci. De plus, cette méthode, étant très générique, peut s'appliquer à une multitude de systèmes fondés sur l'opération de recombinaison, voir aussi les chapitres 7, 8 et 9.

6.1 Description de la méthode

Nous analysons d'abord le calcul dans les systèmes présentés dans les théorèmes 4.3.1 et 5.2.2. L'idée principale de ces calculs est la suivante : le mot codifiant une forme syntaxique de la grammaire est changé à une de ses extrémités en se recombinant avec un axiome, dont nous dirons qu'il est associé à la règle utilisée. Nous sommes intéressés par un seul résultat de cette recombinaison et les règles sont arrangées dans les composants d'une manière qui permette l'élimination du deuxième mot, ainsi que du mot initial dans le cadre des systèmes distribués à changement de phase étendus. Pour préserver l'axiome utilisé, nous faisons appel à des règles spéciales qui conservent les axiomes.

Dans le cadre de cette méthode, une simulation correcte de la grammaire est faite par le choix judicieux de l'emplacement des règles dans les composants. Comme les axiomes nécessaires sont présents tout le temps, n'importe quelle règle d'un composant est utilisable chaque fois qu'on arrive dans ce composant. Or, à chaque fois, il n'y a qu'une seule règle à appliquer et le reste des règles peut poser des problèmes pour une simulation correcte.

C'est pourquoi nous proposons une nouvelle méthode de simulation qui corrige

certains défauts de la méthode ci-dessus. Maintenant, nous ne permettons plus aux axiomes d'être présents à chaque étape, mais uniquement pendant les étapes où ils sont vraiment utilisés. Pour cela, nous les marquons par un nombre, *l'état*, et à chaque étape nous l'incrémentons modulo k. Lorsqu'on arrive à zéro, le bon mot est recréé et peut être utilisé pour la recombinaison. Nous appelons ces mots des molécules assistantes. L'apparition d'une molécule assistante à une étape quelconque dépend donc de sa *période* k et de son état initial.

Ceci étant dit, observons maintenant les changements intervenus au niveau du calcul. L'application d'une règle est contrôlée maintenant non seulement par son emplacement, mais aussi par la période de la molécule assistante associée à cette règle. À chaque étape, seules les règles nécessaires peuvent donc s'appliquer, les autres règles du même composant ne pouvant plus s'appliquer puisque le deuxième mot qui participe à la recombinaison, c'est-à-dire la molécule assistante associée, n'est pas présent. Cela nous donne la possibilité de raffiner le contrôle imposé par les composants et, en réarrangeant les règles, de diminuer le nombre de composants nécessaires pour simuler une grammaire.

Dans ce qui suit nous montrons deux exemples d'utilisation de cette technique où on diminue le nombre de composants nécessaires pour simuler une grammaire arbitraire par les systèmes distribués à changement de phase et par l'extension de ces derniers.

6.2 Systèmes distribués à changement de phase étendus de degré 2

Dans cette section nous montrons comment il est possible d'appliquer les réflexions ci-dessus. Nous considérons les systèmes distribués à changement de phase étendus et nous montrons que deux composants suffisent pour simuler une grammaire arbitraire.

Théorème 6.2.1. *Soit $G = (N, T, P, S)$ une grammaire arbitraire. Il existe un système distribué à changement de phase étendu de degré 2, $E_G = (V, T, A, R_1, R_2)$ qui simule G et pour lequel $L(G) = L(E_G)$.*

Démonstration. Nous allons prouver cette affirmation de la manière suivante. Nous montrerons comment il est possible de simuler les dérivations de la grammaire formelle G et, par conséquent, nous obtiendrons que $L(G) \subseteq L(E_G)$. Dans le même temps, nous montrerons que toutes les autres possibilités de calcul n'aboutissent pas à un mot terminal, ce qui prouvera l'énoncé.

Notre simulation est fondée sur la méthode «faire-tourner-et-simuler», voir aussi la section 4.1.2. Nous avons déjà décrit comment adapter cette méthode aux systèmes distribués à changement de phase étendus au cours de la démonstration du théorème 5.2.2.

Nous donnons d'abord le schéma du calcul ainsi qu'une description de la méthode que nous utilisons, puis nous donnons la définition formelle du système.

Le schéma du calcul

Le calcul dans E_G suit le schéma présenté à la figure 6.1. Les sommets du schéma représentent une configuration de mots pendant le calcul. Nous énumérons toutes les configurations possibles et nous mettons leur numéro dans le coin supérieur droit. Le schéma est composé de deux parties : la partie supérieure et la partie inférieure. Les configurations ayant le même numéro dans les deux parties se produisent au même instant. Il est facile de voir que dans la partie inférieure les indices de X et Y sont décrémentés simultanément, de sorte qu'on continue avec la partie supérieure seulement dans le cas où les indices de X et Y sont tous les deux égaux à un. La partie supérieure complète donc la rotation, effectue la simulation des règles de la grammaire et permet d'obtenir un mot-résultat. Il est facile de vérifier que les mots d'une configuration arrivent toujours dans le composant ayant le même numéro. Nous pouvons donc dire que chaque configuration a un numéro de composant associé. Le symbole **w** dans les configurations est considéré comme une variable et il prend des valeurs différentes dans les différentes configurations. Par exemple, si dans la configuration 8 le symbole **w** est égal à $w'a_i$, dans la configuration 1 sa valeur peut être w'. Nous allons montrer que le calcul suit le schéma de la figure 6.1, c'est-à-dire que tous les mots produits dans une configuration seront éliminés, excepté les mots appartenant à la configuration suivante.

Les molécules assistantes

La transition entre deux configurations est faite en effectuant des changement à une des extrémités du mot. Pour cela le mot est recombiné avec des molécules assistantes de la forme Z_kT_k ou T_mZ_m qui effectuent des changement à l'extrémité droite, respectivement gauche, du mot. La méthode des molécules assistantes nous permet d'éviter les branches erronées du calcul.

Les molécules assistantes, voir aussi la section 6.1, sont créées seulement au moment où elles sont nécessaires. Par exemple, les molécules assistantes pour la configuration 5 (Z_4Y_j''', XZ_{14}, C_1Z_{15}) sont créées à l'étape précédente et elles n'existeront pas dans la configuration 8. Chaque molécule peut être dans l'un des huit états et à chaque étape l'état de la molécule est incrémenté modulo 8. Lorsque l'état de la molécule est zéro, et dans ce cas nous l'omettrons, nous obtenons la molécule assistante qui peut être utilisée ensuite. Par exemple, nous sommes dans la configuration 5 et nous avons la molécule assistante XZ_{14}. Nous la marquons avec 1 à l'aide de la règle $1.z.5$: $(X|Z_{14}, Z|Z_{14}^1) \vdash_{1.z.5} (XZ_{14}^1, ZZ_{14})$. À l'étape suivante la molécule XZ_{14}^1 change son état en 2 à la suite de l'application de la règle $2.z.1$. Nous continuons d'une manière similaire et nous obtenons le calcul suivant : $XZ_{14} \Rightarrow XZ_{14}^1 \Rightarrow XZ_{14}^2 \Rightarrow \ldots \Rightarrow XZ_{14}^7 \Rightarrow XZ_{14}$. Ceci dit, la molécule XZ_{14} apparaît donc avec une période égale à 8 et elle est présente dans les configurations 4, 5 et 6, mais elle n'est utilisable qu'à la configuration 5, voir page 74.

Cette technique permet donc de raffiner le contrôle de l'application des règles du même composant et elle permet leur division en sous-ensembles activés par des molécules assistantes spécifiques.

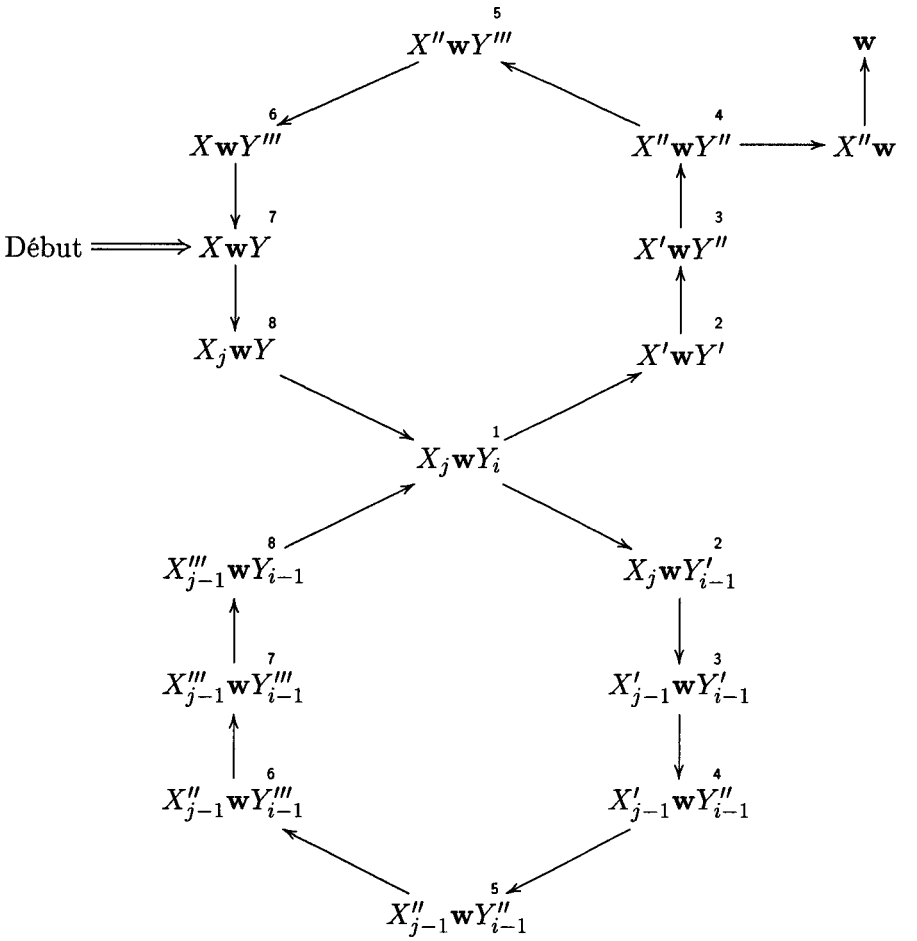


FIG. 6.1 – Le schéma du calcul

La définition formelle du système

Maintenant nous donnons la définition formelle du système.

Posons $E_G = (V, T, A, R_1, R_2)$.

Soient $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(B = a_n)$ et $B \notin V \cup T$.

Dans ce qui suit nous supposons que :
$1 \leq i \leq n,\ 1 \leq j \leq n-1,\ 2 \leq l \leq n,\ 1 \leq k \leq 11, 12 \leq m \leq 20,\ 1 \leq s \leq 7,$
$\mathbf{a}, \mathbf{b} \in N \cup T \cup \{B\}$.

L'alphabet V est défini par
$V = N \cup T \cup \{B\} \cup \{X, Y, X', Y'X'', Y'', Y''', X_i, Y_i, X'_j, Y'_j, X''_j, Y''_j, X'''_j, Y'''_j, Z_k, Z^s_k,$
$Z_m, Z^s_m, Z, C_1, C_2\}$.

L'alphabet terminal T est celui de la grammaire formelle G.

Les axiomes sont définis par :
$A = \{XBSY\} \cup \{X_i a_i Z_{16}, Z_5 Y_j, X'''_j Z^1_{19}, Z^1_9 Y, X Z^2_{14}, C_1 Z^2_{15}, Z^2_4 Y'''_j, X''_j Z^3_{18}, Z^3_7 Y''',$
$Z^3_8 C_2, X'' Z^4_{13}, Z^4_3 Y''_j, X'_j Z^5_{17}, Z^5_6 Y'', Z^6_1 Y'_j, Z^6_2 Y', X' Z^6_{12}, X_j Z^7_{20}, Z^7_{10} Y_i\} \cup \{Z_k Z, Z^s_k Z,$

$ZZ_m, ZZ_m^s, \} \cup \{Z_{11}^7 vY : \exists u \to v \in P\}$.

Maintenant nous allons donner la définition des composants du système. La numérotation des règles n'ayant pas de z dans leur nom suit les conventions suivantes.

- Le premier numéro indique à quel composant appartient la règle.
- Le deuxième numéro indique dans quelle configuration cette règle peut être utilisée.
- Le troisième numéro indique la position sur le schéma du calcul de la molécule à laquelle cette règle peut s'appliquer : s'il est égal a 1, la règle s'applique à une molécule qui est dans une configuration de la partie inférieure du schéma et s'il est plus grand que 1, la règle s'applique à une molécule étant dans une configuration de la partie supérieure du schéma.

La numérotation des règles ayant z dans leur nom suit les conventions suivantes.

- Le premier numéro indique à quel composant appartient la règle.
- Le deuxième numéro, qui est après z, indique la parité de l'étape d'utilisation de la molécule assistante à laquelle cette règle s'applique. Si ce numéro est entre 1 et 8, la règle s'applique à une molécule assistante qui est utilisable pendant une étape impaire ; si le numéro est entre 9 et 16, la règle s'applique à une molécule assistante qui est utilisable pendant une étape paire.

Composant R_1 :

Règles pour la transition entre les configurations :

$$1.1.1 : \frac{\mathbf{a} \mid Y_l}{Z_1 \mid Y'_{l-1}} ; \quad 1.1.2 : \frac{\mathbf{a} \mid Y_1}{Z_2 \mid Y'} ; \quad 1.1.3 : \frac{X_1 \mid \mathbf{a}}{X' \mid Z_{12}} ;$$

$$1.3.1 : \frac{\mathbf{a} \mid Y'_j}{Z_3 \mid Y''_j} ; \quad 1.3.2 : \frac{X' \mid \mathbf{a}}{X'' \mid Z_{13}} ; \quad 1.5.1 : \frac{\mathbf{a} \mid Y''_j}{Z_4 \mid Y'''_j} ;$$

$$1.5.2 : \frac{X'' \mid \mathbf{a}}{X \mid Z_{14}} ; \quad 1.5.3 : \frac{X'' \mid \mathbf{a}}{\varepsilon \mid C_1 Z_{15}} ; \quad 1.7.1 : \frac{\mathbf{a} \mid Y'''_j}{Z_5 \mid Y_j} ; \quad 1.7.2 : \frac{X \mid \mathbf{a}}{X_i a_i \mid Z_{16}} ;$$

Règles pour la création des molécules assistantes :

Molécules assistantes pour les étapes impaires :

$$1.z.1 : \frac{Z_{k_1} \mid T_{k_1}}{Z^1_{k_1} \mid Z} ; \quad 1.z.2 : \frac{Z^2_{k_1} \mid T_{k_1}}{Z^3_{k_1} \mid Z} ; \quad 1.z.3 : \frac{Z^4_{k_1} \mid T_{k_1}}{Z^5_{k_1} \mid Z} ; \quad 1.z.4 : \frac{Z^6_{k_1} \mid T_{k_1}}{Z^7_{k_1} \mid Z} ;$$

$$1.z.5 : \frac{T_{m_1} \mid Z_{m_1}}{Z \mid Z^1_{m_1}} ; \quad 1.z.6 : \frac{T_{m_1} \mid Z^2_{m_1}}{Z \mid Z^3_{m_1}} ; \quad 1.z.7 : \frac{T_{m_1} \mid Z^4_{m_1}}{Z \mid Z^5_{m_1}} ; \quad 1.z.8 : \frac{T_{m_1} \mid Z^6_{m_1}}{Z \mid Z^7_{m_1}} ;$$

Molécules assistantes pour les étapes paires :

$$1.z.9 : \frac{Z^1_{k_2} \mid T_{k_2}}{Z^2_{k_2} \mid Z} ; \quad 1.z.10 : \frac{Z^3_{k_2} \mid T_{k_2}}{Z^4_{k_2} \mid Z} ; \quad 1.z.11 : \frac{Z^5_{k_2} \mid T_{k_2}}{Z^6_{k_2} \mid Z} ; \quad 1.z.12 : \frac{Z^7_{k_2} \mid T_{k_2}}{Z_{k_2} \mid Z} ;$$

$$1.z.13 : \frac{T_{m_2} \mid Z^1_{m_2}}{Z \mid Z^2_{m_2}} ; \quad 1.z.14 : \frac{T_{m_2} \mid Z^3_{m_2}}{Z \mid Z^4_{m_2}} ; \quad 1.z.15 : \frac{T_{m_2} \mid Z^5_{m_2}}{Z \mid Z^6_{m_2}} ; \quad 1.z.16 : \frac{T_{m_2} \mid Z^7_{m_2}}{Z \mid Z_{m_2}} ;$$

Composant R_2 :

Règles pour la transition entre les configurations :

$$2.2.1 : \frac{X_l \mid \mathbf{a}}{X'_{l-1} \mid Z_{17}} ; \quad 2.2.2 : \frac{\mathbf{a} \mid Y'}{Z_6 \mid Y''} ; \quad 2.4.1 : \frac{X'_j \mid \mathbf{a}}{X''_j \mid Z_{18}} ; \quad 2.4.2 : \frac{\mathbf{ab} \mid Y''}{Z_7 \mid Y'''} ;$$

$$2.4.3 : \frac{\mathbf{a} \mid BY''}{Z_8C_2 \mid \varepsilon} ; \quad 2.6.1 : \frac{X''_j \mid \mathbf{a}}{X'''_j \mid Z_{19}} ; \quad 2.6.2 : \frac{\mathbf{a} \mid Y'''}{Z_9 \mid Y} ; \quad 2.8.1 : \frac{X'''_j \mid \mathbf{a}}{X_j \mid Z_{20}} ;$$

$$2.8.2 : \frac{\mathbf{a} \mid a_iY}{Z_{10} \mid Y_i} ; \quad 2.8.3 : \frac{\mathbf{a} \mid uY}{Z_{11} \mid vY} ;$$

Règles pour la création des molécules assistantes :
Molécules assistantes pour les étapes impaires :

$$2.z.1 : \frac{Z^1_{k_1} \mid T_{k_1}}{Z^2_{k_1} \mid Z} ; \quad 2.z.2 : \frac{Z^3_{k_1} \mid T_{k_1}}{Z^4_{k_1} \mid Z} ; \quad 2.z.3 : \frac{Z^5_{k_1} \mid T_{k_1}}{Z^6_{k_1} \mid Z} ; \quad 2.z.4 : \frac{Z^7_{k_1} \mid T_{k_1}}{Z_{k_1} \mid Z} ;$$

$$2.z.5 : \frac{T_{m_1} \mid Z^1_{m_1}}{Z \mid Z^2_{m_1}} ; \quad 2.z.6 : \frac{T_{m_1} \mid Z^3_{m_1}}{Z \mid Z^4_{m_1}} ; \quad 2.z.7 : \frac{T_{m_1} \mid Z^5_{m_1}}{Z \mid Z^6_{m_1}} ; \quad 2.z.8 : \frac{T_{m_1} \mid Z^7_{m_1}}{Z \mid Z_{m_1}} ;$$

Molécules assistantes pour les étapes paires :

$$2.z.9 : \frac{Z_{k_2} \mid T_{k_2}}{Z^1_{k_2} \mid Z} ; \quad 2.z.10 : \frac{Z^2_{k_2} \mid T_{k_2}}{Z^3_{k_2} \mid Z} ; \quad 2.z.11 : \frac{Z^4_{k_2} \mid T_{k_2}}{Z^5_{k_2} \mid Z} ; \quad 2.z.12 : \frac{Z^6_{k_2} \mid T_{k_2}}{Z^7_{k_2} \mid Z} ;$$

$$2.z.13 : \frac{T_{m_2} \mid Z_{m_2}}{Z \mid Z^1_{m_2}} ; \quad 2.z.14 : \frac{T_{m_2} \mid Z^2_{m_2}}{Z \mid Z^3_{m_2}} ; \quad 2.z.15 : \frac{T_{m_2} \mid Z^4_{m_2}}{Z \mid Z^5_{m_2}} ; \quad 2.z.16 : \frac{T_{m_2} \mid Z^6_{m_2}}{Z \mid Z^7_{m_2}} ;$$

où $1 \leq k_1 \leq 5$, $6 \leq k_2 \leq 11$, $12 \leq m_1 \leq 16$, $17 \leq m_2 \leq 20$.
Les symboles T_k et T_m sont donnés par les tableaux suivants :

k	1	2	3	4	5	6	7	8	9	10	11
T_k	Y'_j	Y'	Y''_j	Y'''_j	Y_j	Y''	Y'''	C_2	Y	Y_i	vY

m	12	13	14	15	16	17	18	19	20
T_m	X'	X''	X	C_1	X_ia_i	X'_j	X''_j	X'''_j	X_j

Les composants R_1 et R_2 contiennent aussi les règles suivantes :
$\frac{\alpha \mid \varepsilon}{\alpha \mid \varepsilon}$ où $\alpha \in \{Z_kZ, Z^s_kZ, ZZ_m, ZZ^s_m\}$.

Notations

Nous allons utiliser la même notation que celle utilisée dans le théorème 5.2.2 :
$\frac{w_1 \mid w_2}{w'_1 \mid w'_2} \vdash_r \frac{w_1w'^*_2}{w'_1w_2}$, w_1w_2
La signification de cette notation peut être retrouvée à la page 61.

Description du calcul

Il est facile de voir qu'à chaque configuration n nous avons trois types de molécules assistantes qui sont présentes :

- Les molécules assistantes pour la configuration précédente, $n-1$, car elles ont participé à une recombinaison à l'étape précédente et, faisant partie de σ^* par définition, elles sont présentes pendant cette étape.
- Les molécules assistantes pour la configuration courante, n, car elles ont été créées à l'étape précédente.
- Les molécules assistantes pour l'étape suivante, $n+1$, car, étant créées pendant cette étape, elles peuvent participer tout de suite à une recombinaison à cause de la définition des systèmes distribués à changement de phase étendus.

Le fonctionnement de toutes les configurations est similaire. Nous discuterons maintenant le comportement des configurations 5 et 1, voir la figure 6.1.

La configuration 5. Nous avons les molécules suivantes : $X''wY'''$, $X''_{j-1}wY''_{i-1}$ $(i,j>1)$, ainsi que les molécules assistantes suivantes : XZ_{14}, C_1Z_{15}, $Z_4Y'''_j$, X''_jZ_{18}, Z_7Y''', Z_8C_2, X'''_jZ_{19}, Z_9Y. Les trois premières molécules sont les molécules assistantes pour la configuration 5. Nous sommes dans le premier composant et nous avons le calcul suivant :

$$\frac{X'' \mid wY'''}{X \mid Z_{14}} \vdash_{1.5.2} \frac{XwY'''}{X''Z_{14}}, \quad X''wY''',$$

$$\frac{X'' \mid wY'''}{\varepsilon \mid C_1Z_{15}} \vdash_{1.5.3} \frac{wY'''}{X''C_1Z_{15}}, \quad X''wY''',$$

$$\frac{X''_{j-1}w \mid Y''_{i-1}}{Z_4 \mid Y'''_{i-1}} \vdash_{1.5.1} \frac{X''_{j-1}wY'''_{i-1}}{Z_4Y''_{i-1}}, \quad X''_{j-1}wY''_{i-1}.$$

Les mots XwY''' et $X''_{j-1}wY'''_{i-1}$ sont dans la configuration 6. Les molécules $X''wY'''$, wY''' et $X''_{j-1}wY''_{i-1}$ sont éliminées à l'étape suivante :

$$\frac{X''w \mid Y'''}{Z_9 \mid Y} \vdash_{2.6.2} \frac{X''wY\uparrow}{Z_9Y'''},$$

$$\frac{w \mid Y'''}{Z_9 \mid Y} \vdash_{2.6.2} \frac{wY\uparrow}{Z_9Y'''},$$

$$\frac{X''_{j-1} \mid wY''_{i-1}}{X'''_{j-1} \mid Z_{19}} \vdash_{2.6.1} \frac{X'''_{j-1}wY''_{i-1}\uparrow}{X''_{j-1}Z_{19}}.$$

Nous avons donc montré que nous obtenons les molécules XwY''' et $X''_{j-1}wY'''_{i-1}$ et que les autres molécules sont éliminées, c'est-à-dire que nous avons suivi le schéma du calcul.

La configuration 1. Nous avons les molécules suivantes : X_jwY_i, ainsi que les molécules assistantes suivantes : $Z_1Y'_j$, Z_2Y', $X'Z_{12}$, $Z_{10}Y_i$, X_jZ_{20}, X'_jZ_{17}, Z_6Y''.

Nous allons distinguer 4 cas :

a) $i,j>1$

$$\frac{X_jw \mid Y_i}{Z_1 \mid Y'_{i-1}} \vdash_{1.1.1} \frac{X_jwY'_{i-1}}{Z_1Y_i}, \quad X_jwY_i.$$

La molécule $X_jwY'_{i-1}$ est dans la configuration 2. La molécule X_jwY_i est éliminée pendant l'étape suivante :

$$\frac{X_j \mid wY_i}{X'_{j-1} \mid Z_{17}} \vdash_{2.2.1} \frac{X'_{j-1}wY_i\uparrow}{X_jZ_{17}}.$$

b) $i, j = 1$

Nous pouvons appliquer deux règles : 1.1.2 ou 1.1.3 :

$$\frac{X_1 w \mid Y_1}{Z_2 \mid Y'} \vdash_{1.1.2} \frac{X_1 w Y'}{Z_2 Y_1},$$

$$\frac{X_1 \mid w Y_1}{X' \mid Z_{12}} \vdash_{1.1.3} \frac{X' w Y_1 \uparrow}{X_1 Z_{12}}.$$

Nous pouvons appliquer encore une fois la règle 1.1.3 ou 1.1.2 au résultat de l'application précédente :

$$\frac{X_1 \mid w Y'}{X' \mid Z_{12}} \vdash_{1.1.3} \frac{X' w Y'}{X_1 Z_{12}}, \quad X_1 w Y'.$$

La molécule $X' w Y'$ est dans la configuration 2. La molécule $X_1 w Y'$ sera éliminée à l'étape suivante :

$$\frac{X_1 w \mid Y'}{Z_6 \mid Y''} \vdash_{2.2.2} \frac{X_1 w Y'' \uparrow}{Z_6 Y'}.$$

c) $i > 1, j = 1$

$$\frac{X_1 w \mid Y_i}{Z_1 \mid Y'_{i-1}} \vdash_{1.1.1} \frac{X_1 w Y'_{i-1} \uparrow}{Z_1 Y_i},$$

$$\frac{X_1 \mid w Y_i}{X' \mid Z_{12}} \vdash_{1.1.3} \frac{X' w Y_i \uparrow}{X_1 Z_{12}}.$$

Nous pouvons appliquer encore une fois la règle 1.1.1 ou 1.1.3 aux molécules obtenues plus haut :

$$\frac{X' w \mid Y_i}{Z_1 \mid Y'_{i-1}} \vdash_{1.1.1} \frac{X' w Y'_{i-1} \uparrow}{Z_1 Y_i}.$$

d) $i = 1, j > 1$

$$\frac{X_j w \mid Y_1}{Z_2 \mid Y'} \vdash_{1.1.2} \frac{X_j w Y'}{Z_2 Y_1}, \quad X_j w Y_1.$$

Les molécules $X_j w Y_1$ et $X_j w Y'$ sont éliminées pendant l'étape suivante :

$$\frac{X_j \mid w Y_1}{X'_{j-1} \mid Z_{17}} \vdash_{2.2.1} \frac{X'_{j-1} w Y_1 \uparrow}{X_j Z_{17}},$$

$$\frac{X_j \mid w Y'}{X'_{j-1} \mid Z_{17}} \vdash_{2.2.1} \frac{X'_{j-1} w Y' \uparrow}{X_j Z_{17}},$$

$$\frac{X_j w \mid Y'}{Z_6 \mid Y''} \vdash_{2.2.2} \frac{X_j w Y'' \uparrow}{Z_6 Y'}.$$

Nous pouvons appliquer encore une fois la règle 2.2.1 ou 2.2.2 aux molécules obtenues plus haut :

$$\frac{X_j \mid w Y''}{X'_{j-1} \mid Z_{17}} \vdash_{2.2.1} \frac{X'_{j-1} w Y'' \uparrow}{X_j Z_{17}}.$$

Nous avons donc obtenu les molécules $X_j w Y'_{i-1}$ et $X' w Y'$. Dans le même temps toutes les autres molécules ont été éliminées, c'est-à-dire que nous avons suivi le schéma du calcul.

Remarques finales

D'une manière décrite plus haut nous passons d'une configuration à l'autre en suivant le schéma du calcul de la figure 6.1. La simulation de la règle $u \to v$ de la grammaire est faite à la configuration 8 à l'aide de la règle 2.8.3. Nous avons donc réalisé la méthode «faire-tourner-et-simuler» qui permet une simulation de la grammaire.

Afin d'obtenir le résultat, nous appliquons dans la configuration 4 la règle 2.4.3, puis la règle 1.5.3 :

$$\frac{X''w \mid BY''}{Z_8C_2 \mid \varepsilon} \vdash_{2.4.3} \frac{X''w}{Z_8C_2BY''}, \quad X''wBY''.$$

$$\frac{X'' \mid w}{\varepsilon \mid C_1Z_{15}} \vdash_{1.5.3} \frac{\mathbf{w}\uparrow}{X''C_1Z_{15}}.$$

$$\frac{X'' \mid w}{X \mid Z_{14}} \vdash_{1.5.2} \frac{Xw\uparrow}{X''Z_{14}}.$$

Si $w \in T^*$, il fait partie du résultat. Nous remarquons que les molécules Z_8C_2BY'' et $X''C_1Z_{15}$, une fois produites, persistent, mais elles ne permettent pas de produire de nouveaux mots.

Il est facile de voir qu'en suivant le schéma du calcul nous obtenons tous les mots de $L(G)$, voir aussi la section 4.1.2, et en vue des réflexions présentées plus haut il est clair que le système ne produit pas d'autres mots.

□

6.3 Systèmes distribués à changement de phase de degré 1

Dans cette section nous montrons un autre exemple de l'application de la méthode des molécules assistantes. Nous considérons les systèmes distribués à changement de phase et nous montrons qu'un seul composant suffit pour simuler une grammaire arbitraire.

Théorème 6.3.1. *Soit $G = (N, T, P, S)$ une grammaire arbitraire. Il existe un système distribué à changement de phase de degré 1, $D_G = (V, T, A, R)$ qui simule G et pour lequel $L(G) = L(D_G)$.*

Ce théorème est un corollaire du théorème 4.3.1 et du lemme suivant.

Lemme 6.3.2. *Pour tout système distribué à changement de phase D_G^2 construit comme au théorème 4.3.1 il y a un système distribué à changement de phase de degré 1, $D_G = (V, T, A, R)$ qui simule D_G^2 et pour lequel $L(D_G) = L(D_G^2)$.*

Démonstration. Pour prouver le lemme nous transformons, pour des raisons techniques, le système distribué à changement de phase D_G^2 du théorème 4.3.1, en lui donnant une forme plus approprié pour notre simulation. Pour cela, certaines lettres

sont numérotées et certaines règles ont des sites plus grands, mais ces transformations n'altèrent ni le fonctionnement du système, ni le langage produit. Nous ne faisons plus la distinction entre le système transformé et le système du théorème 4.3.1 et nous nous référons à lui dans ce qui suit par D_G^2, voir figure 6.2. Ce système possède les propriétés suivantes :

- Il a deux composants.
- Il possède deux sous-ensembles indépendants d'axiomes qui persistent pendant le calcul. Un de ces sous-ensembles est utilisé pour les recombinaisons dans le premier composant uniquement, tandis que l'autre est utilisé pour les recombinaisons dans le deuxième composant uniquement.
- Les règles sont faites de telle façon, que les molécules codifiant les différentes étapes de la simulation de la grammaire sont complémentaires seulement avec les axiomes de ces deux sous-ensembles.

Nous allons utiliser la méthode des molécules assistantes pour simuler D_G^2. Premièrement, nous plaçons les règles 1.1 à 1.9 de R_1^2 et 2.1 à 2.8 de R_2^2, que nous appelons règles principales, dans un seul composant. On voit facilement que les règles principales permettent de réaliser la méthode «faire-tourner-et-simuler», tandis que les autres règles permettent de propager les axiomes. Chaque axiome du système initial faisant partie de A_1^2 ou A_2^2, c'est-à-dire associé à une règle principale, devient une molécule assistante de période 2. Comme nous avons un seul état, nous remplaçons son numéro par un prime. On dira que la molécule assistante est à l'état passif lorsqu'elle est primée, sinon on dira qu'elle est à l'état actif.

Maintenant nous organisons le calcul d'une manière à avoir les molécules assistantes de A_1^2 à l'état actif pendant une étape impaire et à l'état passif pendant une étape paire. D'une manière similaire, nous faisons apparaître les molécules assistantes de A_2^2 à l'état passif pendant une étape impaire et à l'état actif pendant une étape paire. Nous observons que les molécules assistantes qui sont à l'état passif ne s'accordent qu'avec les règles qui permettent de les faire passer à l'état actif. Les molécules assistantes qui se trouvent à l'état actif s'accordent avec les règles qui permettent les faire passer à l'état passif, ainsi qu'avec les règles correspondant aux règles principales de D_G^2. Nous observons aussi que les molécules assistantes possèdent leur complément par rapport aux règles de changement d'état.

On voit que par ce procédé nous simulons le comportement du premier composant pendant une étape impaire et le comportement du deuxième composant pendant une étape paire, voir les figures 6.3 et 6.5, ce qui conduit à une simulation correcte du système D_G^2. La transformation du système D_G^2 en D_G est montrée à la figure 6.4.

Par exemple, nous avons la molécule Z_2Y_i à une étape impaire. Elle peut être utilisée pour une recombinaison comme complémentaire par rapport à la règle 2, dans ce cas nous simulons la règle 1.2 du système D_G^2. On peut aussi désactiver cette molécule par la règle $z2$: $(Z_2Y_i, Z_2'R_1) \vdash_{z2} (Z_2'Y_i, Z_2R_1)$. À l'étape suivante qui est paire, la molécule Z_2Y_i n'est plus accessible et la règle 2 ne peut donc pas s'appliquer. D'autre part, nous pouvons activer la molécule $Z_2'R_1$ en utilisant la règle $z2'$, car la molécule complémentaire Z_2R_1 a été produite à l'étape précédente. Nous obtenons donc à nouveau la molécule Z_2R_i et nous sommes à une étape impaire. À

$D_G^2 = (V^2, T^2, A^2, R_1^2, R_2^2)$.

Soient $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(a_n = B)$ et $B \notin N \cup T$.

Dans ce qui suit nous supposons que $1 \leq \mathbf{i} \leq n$, $1 \leq \mathbf{j} \leq n-1$, $2 \leq \mathbf{k} \leq n$, $\mathbf{a}, \mathbf{b} \in N \cup T \cup \{B\}$.

L'alphabet V^2 est défini par

$V^2 = N \cup T \cup \{B\} \cup \{X, Y, X_\mathbf{i}, Y_\mathbf{i}, X'_\mathbf{j}, Y'_\mathbf{j}, X''_\mathbf{j}, Y''_\mathbf{j}, X', Y', X'', Y'', C_1, C_2, D_1, D_2, Z_m\}(1 \leq m \leq 18)$.

L'alphabet terminal T^2 est celui de la grammaire formelle G.

Les axiomes sont définis par : $A^2 = \{XSBY\} \cup A_1^2 \cup A_2^2 \cup \{C_1 Z_{16}, Z_7 C_2, D_1, D_2\}$, où $A_1^2 = \{Z_1 v Y_\mathbf{j}, \exists u \to v a_\mathbf{j} \in P\} \cup \{Z_1 Y_\mathbf{i}, Z_2 Y_\mathbf{i}, Z_3 Y_\mathbf{j}, Z_4 Y'_\mathbf{j}, Z_5 Y''_\mathbf{j}, Z_6 Y, X' Z_8, X'' Z_9\}$ et $A_2^2 = \{X_\mathbf{i} a_\mathbf{i} Z_{11}, X'_\mathbf{j} Z_{13}, X''_\mathbf{j} Z_{14}, Z_{17} Y', Z_{18} Y'', X_{15} Z, X_\mathbf{j} Z_{12}\}$.

Les composants du système sont définis de la manière suivante.

Composant R_1^2 :

$$1.1 : \frac{\mathbf{a} \mid uY}{Z_1 \mid vY_\mathbf{j}}, \exists u \to v a_\mathbf{j} \in P; \quad 1.1' : \frac{\mathbf{a} \mid a_\mathbf{i} u Y}{Z_1 \mid Y_\mathbf{i}}, \exists u \to \varepsilon \in P; \quad 1.2 : \frac{\mathbf{a} \mid a_\mathbf{i} Y}{Z_2 \mid Y_\mathbf{i}};$$

$$1.3 : \frac{\mathbf{ab} \mid Y_\mathbf{k}}{Z_4 \mid Y'_{\mathbf{k}-1}}; \quad 1.4 : \frac{\mathbf{a} \mid Y'_\mathbf{j}}{Z_5 \mid Y''_\mathbf{j}}; \quad 1.5 : \frac{X_1 \mid \mathbf{ab}}{X' \mid Z_8}; \quad 1.6 : \frac{X' \mid \mathbf{a}}{X'' \mid Z_9};$$

$$1.7 : \frac{\mathbf{a} \mid Y''}{Z_6 \mid Y}; \quad 1.8 : \frac{\mathbf{a} \mid BY''}{Z_7 \mid \varepsilon}; \quad 1.9 : \frac{\mathbf{a} \mid Y''_\mathbf{j}}{Z_3 \mid Y_\mathbf{j}}; \quad 1.10 : \frac{C_1 \mid Z_{16}}{\varepsilon \mid D_1};$$

Composant R_2^2 :

$$2.1 : \frac{X \mid \mathbf{a}}{X_\mathbf{i} a_\mathbf{i} \mid Z_{11}}; \quad 2.2 : \frac{X_\mathbf{k} \mid \mathbf{ab}}{X'_{\mathbf{k}-1} \mid Z_{13}}; \quad 2.3 : \frac{X'_\mathbf{j} \mid \mathbf{a}}{X''_\mathbf{j} \mid Z_{14}}; \quad 2.4 : \frac{\mathbf{a} \mid Y_1}{Z_{17} \mid Y'};$$

$$2.5 : \frac{\mathbf{a} \mid Y'}{Z_{18} \mid Y''}; \quad 2.6 : \frac{X'' \mid \mathbf{a}}{X \mid Z_{15}}; \quad 2.7 : \frac{X'' \mid \mathbf{a}}{\varepsilon \mid Z_{16}}; \quad 2.8 : \frac{X''_\mathbf{j} \mid \mathbf{a}}{X_\mathbf{j} \mid Z_{12}}; \quad 2.9 : \frac{Z_7 \mid C_2}{D_2 \mid \varepsilon};$$

Les composants R_1^2 et R_2^2 contiennent aussi les règles suivantes :

$\frac{\alpha \mid \varepsilon}{\alpha \mid \varepsilon}$ pour chaque axiome $\alpha \in A^2$, sauf $XSBY$.

FIG. 6.2 – Le système D_G^2

la figure 6.5 nous pouvons voir une illustration de ce qui a été dit tout à l'heure.

Un raisonnement similaire s'applique à toutes les molécules contenant Z avec des indices.

Les règles 1.8, 1.10, 2.7 et 2.9 sont utilisées en D_G^2 pour obtenir un mot terminal-résultat. En D_G, pour des raisons techniques, elles sont simulées d'une manière spéciale par les règles 7, 16, $x.1$, $x.2$, et $r.1$ à $r.16$.

Description formelle du système

À présent nous donnons une définition formelle de notre système.

Posons $D_G = (V, T, A, R_1)$.

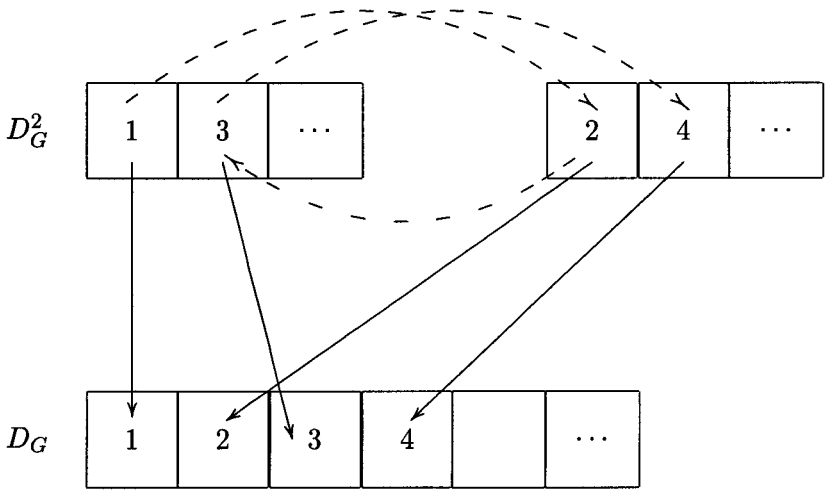


FIG. 6.3 – Simulation des composants du système D_G^2. Dans les cases le numéro de l'étape est indiqué.

Soient $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(B = a_n)$ et $B \notin N \cup T$.

Dans ce qui suit nous supposons que :
$1 \leq i \leq n,\ 1 \leq j \leq n-1,\ 2 \leq k \leq n,\ s \in \{0,2\},\ \mathbf{a}, \mathbf{b} \in N \cup T \cup \{B\}$.

L'alphabet V est défini par $V = \{V^2 \setminus \{C_1, C_2, D_1, D_2\}\} \cup \{R_1, R_2, Z'_{m\ (1 \leq m \leq 18)}\} \cup \{C_1^s, C_2^s, R, Z_7^p, Z_{16}^p, Z_D^p, Z_{E\ (1 \leq p \leq 3)}^p, D_1^s, E_2^s, Z_D, Z_E\}$.

L'alphabet terminal T est celui de D_G^2 $(T = T^2)$, c'est-à-dire celui de la grammaire formelle G.

Les axiomes sont définis par : $A = \{XSBY\} \cup A_1 \cup A_2 \cup A_R$, où

$A_1 = \{Z_1 v Y_i : \exists u \to v a_i \in P\} \cup \{Z_1 Y_i : \exists u \to \epsilon \in P\} \cup$
$\{Z_2 Y_i, Z_3 Y_j, Z_4 Y'_j, Z_5 Y''_j, Z_6 Y, X' Z_8, X'' Z_9\} \cup \{Z'_m R_1 (1 \leq m \leq 7), R_1 Z'_8, R_1 Z'_9\}$.

$A_2 = \{X_i a_i Z'_{11}, X_j Z'_{12}, X'_j Z'_{13}, X''_j Z'_{14}, X Z'_{15}, Z'_{17} Y', Z'_{18} Y''\} \cup$
$\{Z_{17} R_2, Z_{18} R_2, R_2 Z_{m (11 \leq m \leq 16)}\}$.

$A_R = \{C_1^0 Z_{16}, C_1^2 Z_{16}^2, R Z_{16}^1, R Z_{16}^3, Z_7^1 C_2^0,$
$Z_7^3 C_2^2, Z_7^2 R, Z_7 R, D_1^0 Z_D^1, D_1^2 Z_D^3, R Z_D^2, R Z_D, Z_E E_2^0, Z_E^2 E_2^2, Z_E^1 R, Z_E^3 R\}$.

Les axiomes de A_1 et A_2 sont utilisé pour la simulation du comportement du premier et, respectivement, du deuxième composant. Les axiomes de A_R sont utilisés pour l'obtention du résultat.

Simulation des règles du premier composant du système original (le numéro

Le système D_G^2

Composant I

Règle 1.2 : $\dfrac{\mathbf{a} \mid a_iY}{Z_2 \mid Y_i}$

Axiome : Z_2Y_i

Règle pour l'axiome : $\dfrac{Z_2Y_i \mid \varepsilon}{Z_2Y_i \mid \varepsilon}$
(présente dans les deux composants)

Composant II

Règle 2.2 : $\dfrac{X_l \mid \mathbf{ab}}{X'_{l-1} \mid Z_{13}}$

Axiome : X'_jZ_{13}

Règle pour l'axiome : $\dfrac{X'_jZ_{13} \mid \varepsilon}{X'_jZ_{13} \mid \varepsilon}$
(présente dans les deux composants)

Le système D_G (1 composant)

Règle 2 : $\dfrac{\mathbf{a} \mid a_iY}{Z_2 \mid Y_i}$

Axiomes : Z_2Y_i, Z'_2R_1

Règles pour les axiomes : $z2$: $\dfrac{Z_2 \mid Y_i}{Z'_2 \mid R_1}$; $z2'$: $\dfrac{Z'_2 \mid Y_i}{Z_2 \mid R_1}$

Règle 13 : $\dfrac{X_l \mid \mathbf{ab}}{X'_{l-1} \mid Z_{13}}$

Axiomes : X'_jZ_{13}, R_2Z_{13}

Règles pour les axiomes : $z13$: $\dfrac{X'_j \mid Z_{13}}{R_2 \mid Z'_{13}}$; $z13'$: $\dfrac{X'_j \mid Z'_{13}}{R_2 \mid Z_{13}}$

FIG. 6.4 – La transformation du système D_G^2 en D_G.

original est donné entre parenthèses) :

$1(1.1) : \dfrac{\mathbf{a} \mid uY}{Z_1 \mid vY_i}$ $(\exists u \to va_i \in P)$; $1'(1.1') : \dfrac{\mathbf{a} \mid a_iuY}{Z_1 \mid Y_i}$ $(\exists u \to \epsilon \in P)$; $2(1.2) : \dfrac{\mathbf{a} \mid a_iY}{Z_2 \mid Y_i}$;

$3(1.9) : \dfrac{\mathbf{a} \mid Y''_j}{Z_3 \mid Y_j}$; $4(1.3) : \dfrac{\mathbf{ab} \mid Y_k}{Z_4 \mid Y'_{k-1}}$; $5(1.4) : \dfrac{\mathbf{a} \mid Y'_j}{Z_5 \mid Y''_j}$;

$6(1.7) : \dfrac{\mathbf{a} \mid Y''}{Z_6 \mid Y}$; $7(1.8) : \dfrac{\mathbf{a} \mid BY''}{Z_7 \mid C_2^s}$; $8(1.5) : \dfrac{X_1 \mid \mathbf{ab}}{X' \mid Z_8}$; $9(1.6) : \dfrac{X' \mid \mathbf{a}}{X'' \mid Z_9}$;

Simulation des règles du deuxième composant du système original (le numéro original est donné entre parenthèses) :

$11(2.1) : \dfrac{X \mid \mathbf{a}}{X_ia_i \mid Z_{11}}$; $12(2.8) : \dfrac{X''_j \mid \mathbf{a}}{X_j \mid Z_{12}}$; $13(2.2) : \dfrac{X_k \mid \mathbf{ab}}{X'_{k-1} \mid Z_{13}}$;

$14(2.3) : \dfrac{X'_j \mid \mathbf{a}}{X''_j \mid Z_{14}}$; $15(2.6) : \dfrac{X'' \mid \mathbf{a}}{X \mid Z_{15}}$; $16(2.7) : \dfrac{X'' \mid \mathbf{a}}{C_1^s \mid Z_{16}}$;

$17(2.4) : \dfrac{\mathbf{ab} \mid Y_1}{Z_{17} \mid Y'}$; $18(2.5) : \dfrac{\mathbf{a} \mid Y'}{Z_{18} \mid Y''}$;

L'évolution de $X_3a_3wY_3$

Le système D_G^2

Étape impaire (dans le composant 1)

Nous avons Z_4Y_2' et $X_2'Z_{13}$.

$$\frac{X_3a_3w \mid Y_3}{Z_4 \mid Y_2'} \vdash \frac{X_3a_3wY_2'}{Z_4Y_3} \qquad \frac{Z_4Y_2' \mid \varepsilon}{Z_4Y_2' \mid \varepsilon} \vdash \frac{Z_4Y_2'}{Z_4Y_2'} \qquad \frac{X_2'Z_{13} \mid \varepsilon}{X_2'Z_{13} \mid \varepsilon} \vdash \frac{X_2'Z_{13}}{X_2'Z_{13}}$$

Étape paire (dans le composant 2)

$$\frac{X_3 \mid a_3wY_2'}{X_2' \mid Z_{13}} \vdash \frac{X_2'a_3wY_2'}{X_3Z_{13}} \qquad \frac{Z_4Y_2' \mid \varepsilon}{Z_4Y_2' \mid \varepsilon} \vdash \frac{Z_4Y_2'}{Z_4Y_2'} \qquad \frac{X_2'Z_{13} \mid \varepsilon}{X_2'Z_{13} \mid \varepsilon} \vdash \frac{X_2'Z_{13}}{X_2'Z_{13}}$$

Le système D_G

Étape impaire

Nous avons Z_4Y_2', $Z_4'R_1$, $X_2'Z_{13}'$, R_2Z_{13}.

$$\frac{X_3a_3w \mid Y_3}{Z_4 \mid Y_2'} \vdash \frac{X_3a_3wY_2'}{Z_4Y_3} \qquad \frac{Z_4 \mid Y_2'}{Z_4' \mid R_1} \vdash \frac{Z_4'Y_2'}{Z_4R_1} \qquad \frac{X_2' \mid Z_{13}'}{R_2 \mid Z_{13}} \vdash \frac{X_2'Z_{13}}{R_2Z_{13}'}$$

Étape paire

$$\frac{X_3 \mid a_3wY_2'}{X_2' \mid Z_{13}} \vdash \frac{X_2'a_3wY_2'}{X_3Z_{13}} \qquad \frac{Z_4' \mid Y_2'}{Z_4 \mid R_1} \vdash \frac{Z_4Y_2'}{Z_4'R_1} \qquad \frac{X_2' \mid Z_{13}}{R_2 \mid Z_{13}'} \vdash \frac{X_2'Z_{13}'}{R_2Z_{13}}$$

FIG. 6.5 – L'évolution de $X_3a_3wY_3$ dans les deux systèmes.

Création des molécules assistantes de A_1^2 :

Règles de désactivation :

$$z1 : \frac{Z_1 \mid vY_i}{Z_1' \mid R_1} ; \quad z2 : \frac{Z_2 \mid Y_i}{Z_2' \mid R_1} ; \quad z3 : \frac{Z_3 \mid Y_j}{Z_3' \mid R_1} ; \quad z4 : \frac{Z_4 \mid Y_j'}{Z_4' \mid R_1} ;$$

$$z5 : \frac{Z_5 \mid Y_j''}{Z_5' \mid R_1} ; \quad z6 : \frac{Z_6 \mid Y}{Z_6' \mid R_1} ; \quad z8 : \frac{X' \mid Z_8}{R_1 \mid Z_8'} ; \quad z9 : \frac{X'' \mid Z_9}{R_1 \mid Z_9'} ;$$

Règles d'activation :

$$z1' : \frac{Z_1' \mid vY_i}{Z_1 \mid R_1} ; \quad z2' : \frac{Z_2' \mid Y_i}{Z_2 \mid R_1} ; \quad z3' : \frac{Z_3' \mid Y_j}{Z_3 \mid R_1} ; \quad z4' : \frac{Z_4' \mid Y_j'}{Z_4 \mid R_1} ;$$

$$z5' : \frac{Z_5' \mid Y_j''}{Z_5 \mid R_1} ; \quad z6' : \frac{Z_6' \mid Y}{Z_6 \mid R_1} ; \quad z8' : \frac{X' \mid Z_8'}{R_1 \mid Z_8} ; \quad z9' : \frac{X'' \mid Z_9'}{R_1 \mid Z_9} ;$$

Création des molécules assistantes de A_2^2 :

Règles de désactivation :

$$z11' : \frac{X_i a_i \mid Z'_{11}}{R_2 \mid Z_{11}} ; \quad z12' : \frac{X_j \mid Z'_{12}}{R_2 \mid Z_{12}} ; \quad z13' : \frac{X'_j \mid Z'_{13}}{R_2 \mid Z_{13}} ; \quad z14' : \frac{X''_j \mid Z'_{14}}{R_2 \mid Z_{14}} ;$$

$$z15' : \frac{X \mid Z'_{15}}{R_2 \mid Z_{15}} ; \quad z17' : \frac{Z'_{17} \mid Y'}{Z_{17} \mid R_2} ; \quad z18' : \frac{Z'_{18} \mid Y''}{Z_{18} \mid R_2} ;$$

Règles d'activation :

$$z11 : \frac{X_i a_i \mid Z_{11}}{R_2 \mid Z'_{11}} ; \quad z12 : \frac{X_j \mid Z_{12}}{R_2 \mid Z'_{12}} ; \quad z13 : \frac{X'_j \mid Z_{13}}{R_2 \mid Z'_{13}} ; \quad z14 : \frac{X''_j \mid Z_{14}}{R_2 \mid Z'_{14}} ;$$

$$z15 : \frac{X \mid Z_{15}}{R_2 \mid Z'_{15}} ; \quad z17 : \frac{Z_{17} \mid Y'}{Z'_{17} \mid R_2} ; \quad z18 : \frac{Z_{18} \mid Y''}{Z'_{18} \mid R_2} ;$$

Règles pour le résultat :

$$x.1 : \frac{\mathbf{a} \mid C_2^s}{Z_E E_2^s \mid \varepsilon} ; \quad x.2 : \frac{C_1^s \mid \mathbf{a}}{\varepsilon \mid D_1^s Z_D} ;$$

$$r.1 : \frac{C_1^s \mid Z_{16}}{R \mid Z_{16}^1} ; \quad r.2 : \frac{C_1^s \mid Z_{16}^1}{R \mid Z_{16}^2} ; \quad r.3 : \frac{C_1^s \mid Z_{16}^2}{R \mid Z_{16}^3} ; \quad r.4 : \frac{C_1^s \mid Z_{16}^3}{R \mid Z_{16}} ;$$

$$r.5 : \frac{D_1^s \mid Z_D}{R \mid Z_D^1} ; \quad r.6 : \frac{D_1^s \mid Z_D^1}{R \mid Z_D^2} ; \quad r.7 : \frac{D_1^s \mid Z_D^2}{R \mid Z_D^3} ; \quad r.8 : \frac{D_1^s \mid Z_D^3}{R \mid Z_D} ;$$

$$r.9 : \frac{Z_7 \mid C_2^s}{Z_7^1 \mid R} ; \quad r.10 : \frac{Z_7^1 \mid C_2^s}{Z_7^2 \mid R} ; \quad r.11 : \frac{Z_7^2 \mid C_2^s}{Z_7^3 \mid R} ; \quad r.12 : \frac{Z_7^3 \mid C_2^s}{Z_7 \mid R} ;$$

$$r.13 : \frac{Z_E \mid E_2^s}{Z_E^1 \mid R} ; \quad r.14 : \frac{Z_E^1 \mid E_2^s}{Z_E^2 \mid R} ; \quad r.15 : \frac{Z_E^2 \mid E_2^s}{Z_E^3 \mid R} ; \quad r.16 : \frac{Z_E^3 \mid E_2^s}{Z_E \mid R} ;$$

Le fonctionnement et le résultat

Il est facile de voir que chaque règle principale de D_G^2 est simulée directement par une règle de D_G. Nous nous concentrons maintenant sur l'obtention du résultat dans notre système. Dans le système initial les molécules Z' et Z'' ont été utilisées pour enlever les parenthèses X et Y. Ces molécules apparaissaient l'une à une étape paire et l'autre à une étape impaire du calcul. Intuitivement il est clair que pour simuler ce comportement, des molécules assistantes avec une période égale à quatre sont nécessaires. Nous changeons d'abord les parenthèses X et Y en C_1^s et C_2^s pour que la molécule ainsi obtenue ne s'utilise que pour l'obtention du résultat. Puis nous utilisons les molécules assistantes $C_1^s Z_{16}$ et $Z_7 C_2^s$ avec une période égale à quatre qui permettent d'enlever les parenthèses C_1^s et C_2^s. L'indice s, qui représente l'étape de calcul modulo 4, est utilisé pour diminuer le nombre des molécules dans le système. Cette idée vient de l'utilisation du logiciel `TVDHsim`, voir [49], développé par l'auteur pendant son DEA, pour construire le système. Du coup, la diminution du nombre des molécules a permis de contrôler le système obtenu pour dépister les erreurs éventuelles.

Remarques finales

Il est clair qu'en simulant D_G^2 de la manière décrite ci-dessus nous obtenons tous les mots de $L(D_G^2)$ et il est facile de voir que nous ne produisons pas d'autres mots, car pour produire un mot dans notre système nous devons utiliser les règles 7 et 16, ce qui correspond à une production de mot dans D_G^2.

Maintenant, nous parlerons de la complexité de la simulation. Toutes les règles de D_G^2 sauf 1.8, 1.10, 2.7 et 2.9 sont simulées en une étape de calcul, c'est-à-dire que pour simuler une étape de D_G^2 nous utilisons une étape dans notre système. Les règles 1.8, 1.10, 2.7 et 2.9 sont utilisées dans D_G^2 pour obtenir un mot terminal-résultat. Dans D_G nous avons besoins de deux étapes de plus pour obtenir le même résultat.

Nous voulons aussi remarquer qu'à chaque étape l'ensemble des molécules courantes est différent de l'ensemble précédent, c'est-à-dire que $L_k \cap L_{k+1} = \emptyset$. C'est une propriété importante de notre système et elle sera utilisée plus tard. □

6.4 Conclusions

Dans ce chapitre nous avons introduit la méthode des molécules assistantes. Nous avons aussi donné deux exemples d'application de cette méthode en montrant comment il est possible de diminuer le nombre des composants des systèmes distribués à changement de phase et systèmes distribués à changement de phase étendus. Maintenant nous allons présenter d'une manière informelle une abstraction de la méthode des molécules assistantes.

Soit S un système fondé sur la recombinaison, qui est distribué, c'est-à-dire qu'il est constitué à partir de plusieurs composants. Supposons aussi que S permet l'élimination de molécules dans le sens suivant. Pour éliminer une molécule soit on l'enlève du système, soit on l'envoie hors de la portée des règles de son composant. Par exemple, l'élimination d'une molécule dans le cas des systèmes distribués à changement de phase est faite directement, tandis que dans le cas des systèmes de tubes à essai qui seront introduits au chapitre suivant on élimine une molécule d'un composant en l'envoyant à un autre composant. Soient $R_1, \ldots, R_n$ les règles du système distribués dans n composants. Soient $A_1, \ldots, A_n$ les axiomes correspondant aux composants. Maintenant soient $R'_1 \subseteq R_1, \ldots, R'_n \subseteq R_n$ et $A'_1 \subseteq A_1, \ldots, A'_n \subseteq A_n$ tels que pour chaque axiome $x = x_1u_3u_4x_2$ de A'_i, il existe une règle r dans R'_i avec un site qui s'accorde avec cet axiome : $r = u_1\#u_2\$u_3\#u_4$ et il existe une règle $x\#\varepsilon\$x\#\varepsilon$ dans $R_i \setminus R'_i$. En utilisant la méthode des molécules assistantes nous pouvons raffiner le contrôle du système S. Pour cela nous transformons les axiomes de A'_i en molécules assistantes apparaissant avec une certaine période. La création de ces molécules liée à l'étape du calcul doit être dirigée par l'un des procédés suivants :

- Création par numérotation et élimination.
- Création directement par le contrôle.
- Création par imbrication.

Le premier procédé est réalisable s'il est possible d'éliminer certaines molécules à

certains moments du calcul. La molécule assistante possède un numéro, son état, et à chaque étape cet état est incrémenté modulo k. Pour cela il faut pouvoir incrémenter l'état de la molécule, ainsi qu'éliminer la molécule de l'état précédent. La preuve du théorème 6.2.1 est un exemple d'utilisation de ce procédé.

Le deuxième procédé permet de réaliser l'apparition des molécules assistantes pendant certaines étapes directement par le contrôle du système. Au chapitre suivant nous donnons plusieurs preuves utilisant ce procédé.

Le troisième procédé est utilisable si on considère des modèles où il est possible d'avoir des applications itératives des règles de recombinaison pendant la même étape. Dans ce cas, nous pouvons avoir deux niveaux de molécules assistantes. Les molécules assistantes du premier niveau sont crées à l'aide des molécules assistantes du deuxième niveau, ces-derniers pouvant être crées par l'un des procédés ci-dessus. Le théorème 7.2.2 au chapitre suivant utilise ce procédé.

Si la création des molécules assistantes liée à l'étape du calcul est réalisable, le système résultant possédera un contrôle supplémentaire lié aux périodes des molécules assistantes, ce qui permettra l'application de certaines règles à des instants précis. Des exemples d'application de cette technique à différents types de systèmes distribués fondés sur la recombinaison sont donnés aux chapitres suivants.

Chapitre 7

Systèmes de tubes à essai avec filtres alternants

Dans ce chapitre nous introduisons un autre modèle distribué de calcul fondé sur la recombinaison : les systèmes de tubes à essai. C'est l'un des premiers modèles distribués fondés sur la recombinaison qui a été considéré et il est très étudié. L'inspiration pour les systèmes de tubes à essai vient à la fois des systèmes de Head et des grammaires formelles distribuées. En effet, ces systèmes se composent d'un nombre fini de tubes qui sont des systèmes de Head avec des filtres additionnels. Le fonctionnement d'un tel système comporte une étape de calcul et une étape de communication qui sont synchronisées. La distribution des résultats du calcul pendant l'étape de communication est faite selon des règles spécifiques inspirées des règles similaires des grammaires distribuées. Les systèmes de tubes à essai ont un fonctionnement complexe et une multitude de résultats concernant leur puissance d'expression sont connus. Par exemple, trois tubes suffisent pour produire tous les langages récursivement énumérables. D'autre part, la puissance d'expression des systèmes de tubes à essai avec deux tubes est un problème ouvert à présent. Ce problème a suscité des modifications des systèmes de tubes à essai. Ces modifications, intervenant au niveau de la communication, permettent d'obtenir la puissance de calcul d'une machine de Turing avec deux tubes uniquement. Nous proposons aussi une nouvelle variante de systèmes de tubes à essai : les systèmes de tubes à essai avec filtres alternants où le processus de filtrage, utilisé pour la communication, est modifié. Cette modification, permet d'obtenir une grande puissance de calcul avec deux tubes seulement. De plus, il est possible de placer les règles dans le premier tube d'une manière qui permette de ne plus avoir de règles dans le deuxième tube qui est utilisé seulement comme une poubelle. Nous montrons aussi comment il est possible d'appliquer la méthode des molécules assistantes à ces systèmes, malgré le fait qu'ils ne comportent pas d'élimination des molécules.

7.1 Les systèmes de tubes à essai

Définition 7.1.1. [5] Un *système de n tubes à essai* est la construction suivante :

$$\Delta = (V, T, (A_1, R_1, F_1), \ldots, (A_n, R_n, F_n)),$$

où V est un alphabet, $T \subseteq V$ est l'alphabet terminal, $A_i \subseteq V^*$ sont des ensembles finis d'axiomes, R_i, sont des ensembles finis de règles de recombinaison et où $F_i \subseteq V$, les filtres, qui sont des ensembles finis de lettres, $F_i \subseteq V$, $1 \leq i \leq n$.

Chaque triplet (A_i, R_i, F_i), $1 \leq i \leq n$ est appelé *tube à essai*, ou simplement tube, de Δ. Nous pouvons l'appeler aussi *composant.*

Une *configuration* du système est le n-uplet $(L_1, \ldots, L_n)$, $L_i \subseteq V^*$, $1 \leq i \leq n$; L_i est appelé aussi le *contenu* du $i^{\text{ième}}$ tube. Nous définissons la transition entre deux configurations $(L_1, \ldots, L_n)$ et $(L'_1, \ldots, L'_n)$ de la manière suivante :

$(L_1, \ldots, L_n) \Rightarrow (L'_1, \ldots, L'_n)$ ssi

$$L'_i = \left(\bigcup_{j=1}^{n} \sigma_j^*(L_j) \cap F_i^* \right) \cup \left(\sigma_i^*(L_i) \cap \left(V^* - \bigcup_{k=1}^{n} F_k^* \right) \right)$$

où $\sigma_i = (V, R_i)$ est le schéma de recombinaison associé au $i^{\text{ième}}$ tube du système.

En termes usuels, ceci signifie que le calcul consiste en deux micro-étapes répétées itérativement. Pendant la première micro-étape, celle du calcul, chaque tube évolue comme un système de Head ordinaire. Pendant la deuxième micro-étape, celle de la communication, les molécules présentes dans chacun des tubes sont envoyées vers tous les tubes, et les molécules qui franchissent le filtre d'un tube, c'est-à-dire qui sont composées uniquement des lettres qui forment l'ensemble du filtre, restent dans le tube correspondant et forment le langage initial pour l'étape suivante du calcul. Si une molécule peut franchir plusieurs filtres, chacun des tubes correspondants reçoit une copie de cette molécule. Les molécules ne pouvant franchir aucun filtre restent dans le tube dans lequel elles ont été produites. Ce protocole de communication est inspiré de la théorie des grammaires formelles distribuées et il est susceptible de changer lorsqu'on considère des variantes différentes de systèmes de tubes à essai.

Le langage produit par un système de tubes à essai Δ consiste en tous les mots produits par le système qui sont dans le premier tube et qui font partie de l'alphabet terminal.

$L(\Delta) = \{w \in T^* | w \in L_1, \exists L_1, \ldots, L_n \subseteq V^* : (A_1, \ldots, A_n) \stackrel{*}{\Rightarrow} (L_1, \ldots, L_n)\}$.

Nous observons que, par définition, les systèmes de tubes à essai ne possèdent pas d'élimination de molécules, c'est-à-dire qu'une fois produites elles existent toujours. Par contre, une molécule peut quitter le tube dans lequel elle a été produite si elle peut franchir le filtre d'un autre tube et si, dans le même temps, elle ne peut pas franchir le filtre du tube dans lequel elle a été obtenue.

Nous notons par TT_n la famille des langages produits par les systèmes de tubes à essai ayant au plus n tubes.

Il est facile de voir que les systèmes de tubes à essai avec un seul tube sont, en effet, des systèmes de Head étendus. On obtient donc le résultat suivant.

Théorème 7.1.1. $TT_1 = REG$.

Si on considère plusieurs tubes, on obtient une grande puissance de calcul. Trois tubes suffisent déjà pour produire tous les langages récursivement énumérables, voir [34].

Théorème 7.1.2. *[34]* $TT_3 = RE$.

Quant aux systèmes de tubes à essai avec deux composants, le problème de leur puissance de calcul est ouvert, mais, comme le montre l'exemple ci-dessous, ces systèmes peuvent produire des langages qui ne sont pas rationnels.

Exemple 7.1.1.

Nous reprenons l'exemple donné dans [5] et [44].
Considérons le système suivant.

$$\Gamma = (\{a,b,c,d,e\}, \{a,b,c\}, (A_1, R_1, V_1), (A_2, R_2, V_2)),$$
$$A_1 = \{cabc, ebd, dae\},$$
$$R_1 = \{b\#c\$e\#bd,\ da\#e\$c\#a\},$$
$$V_1 = \{a,b,c\},$$
$$A_2 = \{ec, ce\},$$
$$R_2 = \{b\#d\$e\#c,\ c\#e\$d\#a\},$$
$$V_2 = \{a,b,d\}.$$

Les seules recombinaisons possibles dans le premier tube sont les suivantes :

$$(x_1 b|c, e|bd) \vdash_1 (x_1 bbd, ec), \quad \text{pour } x_1 \in \{a,b,c,d,e\}^*,$$
$$(da|e, c|ax_2) \vdash_2 (daax_2, ce), \quad \text{pour } x_2 \in \{a,b,c,d,e\}^*.$$

On voit que ces applications permettent d'ajouter un a, respectivement b, aux mots de la forme cax_2, respectivement $x_1 bc$, en remplaçant dans le même temps c par d. Soit maintenant $ca^n b^m c$, $n, m \geq 1$ un mot dans le premier composant (initialement $n = m = 1$). En utilisant les règles du premier tube nous pouvons produire le mot $da^{n+1} b^{m+1} d$ qui est envoyé dans le deuxième tube. Nous observons que les deux règles de R_1 doivent être utilisées, sinon le mot contient le symbole c et ne peut pas franchir le filtre du deuxième tube.

Dans le deuxième composant nous avons les applications suivantes :

$$(\alpha a^i b^j | d, e|c) \vdash_1 (\alpha a^i b^j c, ed), \quad \alpha \in \{c,d\},$$
$$(c|e, d|a^i b^j \alpha) \vdash_2 (ca^i b^j \alpha, de), \quad \alpha \in \{c,d\}.$$

Seul le mot $ca^i b^j c$ est envoyé dans le premier tube.
Le processus ci-dessus peut être itéré et, par conséquent, on obtient :

$$L(\Gamma) = \{ca^n b^n c \mid n \geq 1\},$$

qui n'est pas un langage rationnel.

On a donc le résultat suivant :

Proposition 7.1.3. $TT_2 \setminus REG \neq \emptyset$.

Les auteurs de [5] et [44] supposent que la famille des langages produits par les systèmes de tubes à essai avec deux tubes est incluse dans la famille des langages algébriques.

7.2 Les systèmes de tubes à essai avec filtres alternants

Définition 7.2.1. Un *système de n tubes à essai avec m filtres alternants* est le $n+2$-uplet suivant :

$$\Gamma = \left(V, T, \left(A_1, R_1, F_1^{(1)}, \ldots, F_1^{(m)}\right), \ldots, \left(A_n, R_n, F_n^{(1)}, \ldots, F_n^{(m)}\right)\right),$$

où V, T, A_i et R_i sont définis comme dans le cas des systèmes de tubes à essai. À la place du filtre F_i pour chaque tube i, se trouve un m-uplet de filtres $F_i^{(r)}$, $1 \leq r \leq m$, dans lequel chaque filtre $F_i^{(r)}$ est défini comme dans le cas précédent.

À chaque étape $k \geq 1$, seul le filtre $F_i^{(r)}$, $r = (k-1) \pmod m + 1$, est actif, c'est-à-dire utilisable, pour le tube i.

Nous définissons la transition entre deux configurations de la manière suivante :

$$\left(L_1^{(1)}, \ldots, L_n^{(1)}\right) = (A_1, \ldots, A_n)$$

$$L_i^{(k+1)} = \left(\bigcup_{j=1}^{n} \sigma_j^*\left(L_j^{(k)}\right) \cap F_i^{(r)*}\right) \cup \left(\sigma_i^*\left(L_i^{(k)}\right) \cap \left(V^* - \bigcup_{k=1}^{n} F_k^{(r)*}\right)\right)$$

où $\sigma_i = (V, R_i)$ est le schéma de recombinaison associé au $i^{\text{ième}}$ tube du système.

Les systèmes obtenus sont très similaires aux systèmes de tubes à essai. La seule différence réside dans le fait qu'à la place d'un filtre unique F_i nous avons un m-uplet de filtres $F_i^{(1)} \ldots F_i^{(m)}$ et le filtre à utiliser dépend du nombre d'étapes qui ont été faites.

Le langage produit par Γ est le suivant :
$L(\Gamma) = \{w \in T^* | \exists k : w \in L_1^{(k)}\}$.

Nous disons que Γ produit le langage vide si pour tout k, l'ensemble $L_1^{(k)}$ ne contient aucun mot terminal.

Nous notons par $TTF_{n,m}$ la famille des langages produits par les systèmes de tubes à essai avec des filtres alternant ayant au plus n tubes et m filtres.

7.2.1 La puissance d'expression de $\text{TTF}_{2,2}$

Dans cette section nous allons montrer un système de tubes à essai ayant deux composants et deux filtres et qui simule une grammaire arbitraire. De plus, les deux filtres du premier tube coïncident.

Théorème 7.2.1. *Soit $G = (N, T, P, S)$ une grammaire arbitraire. Il existe un système de tubes à essai avec filtres alternants ayant deux tubes et deux filtres, $\Gamma = \left(V, T, \left(A_1, R_1, F_1^{(1)}, F_1^{(2)}\right), \left(A_2, R_2, F_2^{(1)}, F_2^{(2)}\right)\right)$ qui simule G et pour lequel on a $L(\Gamma) = L(G)$. De plus, $F_1^{(1)} = F_1^{(2)}$.*

Démonstration. Nous construisons Γ de la manière suivante.

Soient $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(B = a_n)$ et $B \notin N \cup T$.

Dans ce qui suit nous supposons que :

$1 \leq i \leq n, \mathbf{a} \in N \cup T \cup \{B\}, \mathbf{b} \in N \cup T \cup \{B\} \cup \{\diamond\}, \gamma \in \{\alpha, \beta\}$.

L'alphabet V est défini par :

$V = N \cup T \cup \{B\} \cup \{\alpha, \beta\} \cup \{\diamond\} \cup \{X, Y, X_\alpha, X_\beta, X'_\alpha, Y_\alpha, Y'_\alpha, Y_\beta, Z, Z'\}$.

L'alphabet terminal T est celui de la grammaire formelle G.

Les tubes du système sont définis de la manière suivante.

Tube I :

Règles de R_1 :

$$1.1 : \frac{\varepsilon \mid uY}{Z \mid vY}; \qquad 1.2 : \frac{\mathbf{a} \mid a_iY}{Z \mid \beta\alpha^{i-1}Y'_\alpha}; \qquad 1.3 : \frac{\mathbf{a} \mid BY}{Z \mid \diamond Y_\beta};$$

$$1.4 : \frac{X \mid \mathbf{a}}{X_\alpha\beta \mid Z}; \qquad 1.5 : \frac{X \mid \mathbf{a}}{X_\beta\beta\beta \mid Z}; \qquad 1.6 : \frac{\mathbf{b} \mid \beta Y_a}{Z \mid Y_\beta};$$

$$1.7 : \frac{X'_\alpha \mid \alpha}{X_\alpha \mid Z}; \qquad 1.8 : \frac{X'_\alpha \mid \alpha}{X_\beta\beta \mid Z}; \qquad 1.9 : \frac{\gamma \mid \alpha Y_\alpha}{Z \mid Y'_\alpha};$$

Filtres :

$F_1 = F_1^{(1)} = F_1^{(2)} = N \cup T \cup \{B\} \cup \{\alpha, \beta\} \cup \{\diamond\} \cup \{X, Y, X'_\alpha, Y_\alpha\}$.

Axiomes de A_1 :

$\{XBSY, Z\beta\alpha^iY'_\alpha, Z \diamond Y_\beta, X_\alpha\beta Z, X_\beta\beta\beta Z, ZY_\beta, X_\alpha Z, X_\beta\beta Z, ZY'_\alpha\} \cup$
$\{ZvY : \exists u \to v \in P\}$.

Tube II :

Règles de R_2 :

$$2.1 : \frac{\gamma \mid Y'_\alpha}{Z \mid Y_\alpha}; \qquad 2.2 : \frac{X_\alpha \mid \gamma}{X'_\alpha\alpha \mid Z}; \qquad 2.3 : \frac{a_i \mid Y_\beta}{Z \mid Y};$$

$$2.4 : \frac{X_\beta\beta\alpha^i\beta \mid \mathbf{a}}{Xa_i \mid Z}; \qquad 2.5 : \frac{X_\beta\beta\beta \mid \mathbf{a}}{\varepsilon \mid Z'}; \qquad 2.6 : \frac{\mathbf{a} \mid \diamond Y_\beta}{Z' \mid \varepsilon};$$

Filtres :

$F_2^{(1)} = N \cup T \cup \{B\} \cup \{\alpha, \beta\} \cup \{\diamond\} \cup \{X_\alpha, Y'_\alpha\}$.

$F_2^{(2)} = N \cup T \cup \{B\} \cup \{\alpha, \beta\} \cup \{\diamond\} \cup \{X_\beta, Y_\beta\}$.

Axiomes de A_2 :

$\{ZY_\alpha, X'_\alpha\alpha Z, ZY, Xa_iZ, Z'\}$.

Nous affirmons que $L(\Gamma) = L(G)$.

Nous allons prouver cette affirmation de la manière suivante. Nous montrerons comment il est possible de simuler les dérivations de la grammaire formelle G et, par conséquent, nous obtiendrons que $L(G) \subseteq L(\Gamma)$. Dans le même temps, nous

montrerons que toutes les autres possibilités de calcul n'aboutissent pas à un mot terminal, ce qui prouvera l'énoncé.

Notre simulation est fondée sur la méthode «faire-tourner-et-simuler», voir aussi la section 4.1.2. Nous utilisons une technique similaire à celle qui a été utilisée dans [34, 42, 44], voir aussi les théorèmes 4.3.1 et 5.2.2. Nous faisons tourner le mot Xwa_iY d'une lettre en trois étapes. Premièrement, nous codons a_i par $\beta\alpha^i$ et nous obtenons $X_\alpha\beta w\beta\alpha^{i-1}Y'_\alpha$. Puis, nous transférons les α de l'extrémité droite du mot à son extrémité gauche. Finalement, nous décodons $\beta\alpha^i\beta$ en a_i et nous obtenons Xa_iwY. Plus précisément, nous commençons avec le mot Xwa_iY dans le premier tube. Le deuxième tube reçoit le mot $X_\alpha\beta w\beta\alpha^{i-1}Y'_\alpha$ du premier composant. À partir de cet instant, le système fonctionne dans une boucle où un α est inséré à l'extrémité gauche du mot et dans le même temps un α est effacé de son extrémité droite. La boucle se termine lorsqu'il n'y a plus d'α à l'extrémité droite. Après cela, nous obtenons le mot $X_\beta\beta\alpha^i\beta wY_\beta$ dans le deuxième composant et dans ce mot nous pouvons décoder a_i en obtenant Xa_iwY.

Si nous avons le mot $XwBY$, il est possible d'enlever les symboles X, Y et B en produisant w qui, s'il est un mot terminal, fait partie du résultat.

Notations

Nous observons que le site inférieur de chaque règle représente une molécule qui est déjà présente dans le tube correspondant. De même, un des résultats de la recombinaison contiendra Z et ne pourra plus participer à la production du résultat. Nous omettrons donc ces molécules et nous écrivons :

$Xwa_iY \underset{1.2}{\longrightarrow} Xw\beta\alpha^{i-1}Y'_\alpha$ à la place de

$(Xw|a_iY, Z|\beta\alpha^{i-1}Y'_\alpha) \vdash_{1.2} (Xw\beta\alpha^{i-1}Y'_\alpha, Za_iY)$

où par | nous avons souligné les sites de recombinaison. Dans ce qui suit nous marquons les molécules qui peuvent évoluer dans le même tube par ⓗ et nous marquons les molécules qui doivent être envoyées au premier, respectivement deuxième, tube par ①, respectivement ②. Nous écrivons aussi $m\uparrow$ si la molécule m ne peut s'accorder avec aucune règle du tube dans lequel elle se trouve et si, dans le même temps, elle ne peut pas être envoyée à l'autre tube.

Nous allons montrer l'évolution des mots de la forme Xwa_iY. Nous indiquons les règles de recombinaison qui ont été utilisées, ainsi que les molécules résultantes. Nous remarquons, que dans notre système, grâce au parallélisme, plusieurs molécules de cette forme coexistent avec des formes intermédiaires qui évoluent en parallèle. Ces molécules n'interagissent pas les unes avec les autres et nous nous concentrons sur une seule évolution de ce type.

Rotation

Nous montrons comment faire tourner d'une lettre le mot Xwa_iY se trouvant dans le premier tube.

Étape 1.

Tube I.

Xwa_iY ⓗ $\underset{1.2}{\longrightarrow} Xw\beta\alpha^{i-1}Y'_\alpha$ ⓗ $\underset{1.4}{\longrightarrow} X_\alpha\beta w\beta\alpha^{i-1}Y'_\alpha$ ②.

Nous pouvons utiliser ces deux règles dans l'ordre inverse, ce qui donne le même résultat. Nous pouvons aussi utiliser la règle 1.5 à la place de la dernière application. Ceci produit :

$Xw\beta\alpha^{i-1}Y'_\alpha$ ⓗ $\underset{1.5}{\longrightarrow} X_\beta\beta\beta w\beta\alpha^{i-1}Y'_\alpha$ ↑.

Nous pouvons avoir aussi l'application suivante :

Xwa_iY ⓗ $\underset{1.5}{\longrightarrow} X_\beta\beta\beta wa_iY$ ⓗ.

La molécule $X_\alpha\beta w\beta\alpha^{i-1}Y'_\alpha$ est envoyée au deuxième tube, car elle peut franchir le filtre $F_2^{(1)}$; les autres molécules ne peuvent franchir aucun filtre du deuxième tube.

Comme nous l'avons dit avant, il y a d'autres applications des règles de recombinaison sur d'autres molécules, mais ces applications ne sont pas importantes pour notre évolution. Nous ne mentionnerons plus ce fait dans le futur.

Tube II.

Comme nous l'avons dit avant, il y a d'autres applications des règles de recombinaison sur d'autres molécules, mais ces applications ne sont pas importantes pour notre évolution. Nous ne mentionnerons plus ce fait dans le futur.

Étape 2.

Tube II.

$X_\alpha\beta w\beta\alpha^{i-1}Y'_\alpha$ ⓗ $\underset{2.1}{\longrightarrow} X_\alpha\beta w\beta\alpha^{i-1}Y_\alpha$ ⓗ $\underset{2.2}{\longrightarrow} X'_\alpha\alpha\beta w\beta\alpha^{i-1}Y_\alpha$ ①.

Seule la molécule $X'_\alpha\alpha\beta w\beta\alpha^{i-1}Y_\alpha$ peut franchir le filtre F_1.

De cette manière nous avons transféré un α de l'extrémité droite du mot à son extrémité gauche.

Étape 3.

Tube I.

$X'_\alpha\alpha\beta w\beta\alpha^{i-1}Y_\alpha$ ⓗ $\underset{1.9}{\longrightarrow} X'_\alpha\alpha\beta w\beta\alpha^{i-2}Y'_\alpha$ ⓗ $\underset{1.7}{\longrightarrow} X_\alpha\alpha\beta w\beta\alpha^{i-2}Y'_\alpha$ ②.

Nous pouvons avoir aussi les applications suivantes :

$X'_\alpha\alpha\beta w\beta\alpha^{i-1}Y_\alpha$ ⓗ $\underset{1.8}{\longrightarrow} X_\beta\beta\alpha\beta w\beta\alpha^{i-1}Y_\alpha$ ⓗ $\underset{1.9}{\longrightarrow} X_\beta\beta\alpha\beta w\beta\alpha^{i-2}Y'_\alpha$ ↑.

Seul le mot $X_\alpha\alpha\beta w\beta\alpha^{i-1}Y'_\alpha$ peut franchir le filtre $F_2^{(1)}$.

Étape 4.

Tube II.

$X_\alpha\alpha\beta w\beta\alpha^{i-2}Y'_\alpha$ ⓗ $\underset{2.1}{\longrightarrow} X_\alpha\alpha\beta w\beta\alpha^{i-2}Y_\alpha$ ⓗ $\underset{2.2}{\longrightarrow} X'_\alpha\alpha\alpha\beta w\beta\alpha^{i-2}Y_\alpha$ ①.

Seule la molécule $X'_\alpha\alpha\alpha\beta w\beta\alpha^{i-2}Y_\alpha$ peut franchir le filtre F_1.

De cette manière nous avons transféré encore un α de l'extrémité droite du mot à son extrémité gauche. Nous pouvons continuer jusqu'à ce que nous obtenions pour la première fois $X'_\alpha\alpha^i\beta w\beta Y_\alpha$ dans le deuxième tube à une certaine étape p. Cette molécule est envoyée au premier tube où nous pouvons utiliser la règle 1.6 en complétant la rotation de a_i.

Étape p+1.

Tube I.

$X'_\alpha\alpha^i\beta w\beta Y_\alpha$ ⓗ $\underset{1.6}{\longrightarrow} X'_\alpha\alpha^i\beta wY_\beta$ ⓗ $\underset{1.8}{\longrightarrow} X_\beta\beta\alpha^i\beta wY_\beta$ ②.

Nous pouvons avoir aussi les applications suivantes :
$X'_\alpha\alpha^i\beta w\beta Y_\alpha \ \textcircled{h} \underset{1.7}{\longrightarrow} X_\alpha\alpha^i\beta w\beta Y_\alpha \ \textcircled{h} \underset{1.6}{\longrightarrow} X_\alpha\alpha^i\beta wY_\beta \uparrow$.

Seul le mot $X_\beta\beta\alpha^i\beta wY_\beta$ peut franchir le filtre $F_2^{(2)}$. Nous observons aussi que p+1 est un nombre impair. La molécule $X_\beta\beta\alpha^i\beta wY_\beta$ restera donc pour une étape de plus dans le premier composant, car le filtre utilisé dans le deuxième composant pendant une étape impaire est $F_2^{(1)}$. À l'étape suivante, p+2, cette molécule, qui reste inchangée, sera envoyée au deuxième tube, car le nouveau filtre $F_2^{(2)}$ permet son passage.

Étape p+3.

Tube II.

$X_\beta\beta\alpha^i\beta wY_\beta \ \textcircled{h} \underset{2.4}{\longrightarrow} Xa_iwY_\beta \ \textcircled{h} \underset{2.3}{\longrightarrow} Xa_iwY \ \textcircled{1}$.

Le mot Xa_iwY est envoyé dans le premier tube.

On voit qu'on a fini la rotation de a_i.

Simulation des productions de la grammaire

Si nous avons la molécule $XwuY$ dans le premier composant, nous pouvons appliquer la règle 1.1 de notre système pour simuler la règle $u \to v$ de la grammaire.

Obtention du résultat

Nous allons montrer maintenant comment il est possible d'obtenir le résultat. Si nous avons une molécule de la forme $XwBY$, nous pouvons effectuer la rotation de B comme il est décrit plus haut. Nous pouvons aussi enlever les parenthèses X et Y. Dans ce cas, si $w \in T^*$, il fera partie du résultat. Nous allons montrer comment il est possible d'enlever les parenthèses. Supposons que $XwBY$ apparaît pour la première fois dans le composant un à une étape quelconque q. Il est facile de vérifier que q est pair.

Étape q.

Tube 1.

$XwBY \ \textcircled{h} \underset{1.3}{\longrightarrow} Xw \diamond Y_\beta \ \textcircled{h} \underset{1.5}{\longrightarrow} X_\beta\beta\beta w \diamond Y_\beta \ \textcircled{2}$.

Nous pouvons avoir aussi l'application suivante :
$Xw \diamond Y_\beta \ \textcircled{h} \underset{1.4}{\longrightarrow} X_\alpha\beta w \diamond Y_\beta \uparrow$.

Le mot $X_\beta\beta\beta w \diamond Y_\beta$ peut franchir le filtre $F_2^{(2)}$.

Étape q+1.

Tube II.

$X_\beta\beta\beta w \diamond Y_\beta \ \textcircled{h} \underset{2.5}{\longrightarrow} w \diamond Y_\beta \ \textcircled{h} \underset{2.6}{\longrightarrow} w \ \textcircled{1}$.

Si $w \in T^*$, il fait partie du résultat.

□

7.2.2 $\mathrm{TTF}_{2,4}$ avec poubelle

Dans cette section nous présentons un système de tubes à essai avec filtres alternants qui possède deux composants et quatre filtres. En outre, ce système ne contient aucune règle dans le deuxième composant qui est utilisé seulement comme une poubelle. Nous ajoutons aussi que les quatre filtres du deuxième composant sont identiques.

Théorème 7.2.2. *Soit $G = (N, T, P, S)$ une grammaire arbitraire. Il existe un système de tubes à essai avec filtres alternants ayant deux tubes et quatre filtres, $\Gamma = \left(V, T, \left(A_1, R_1, F_1^{(1)}, \ldots, F_1^{(4)}\right), \left(A_2, R_2, F_2^{(1)}, \ldots, F_2^{(4)}\right)\right)$ qui simule G et pour lequel $L(\Gamma) = L(G)$. De plus, le deuxième composant de Γ ne contient aucune règle, c'est-à-dire que $R_2 = \emptyset$, et aussi $F_2^{(1)} = \cdots = F_2^{(4)}$.*

Démonstration. Nous construisons Γ de la manière suivante.

Soient $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(B = a_n)$ et $B \notin N \cup T$.

Dans ce qui suit nous supposons que :
$1 \leq i \leq n, 1 \leq j \leq 4, \mathbf{a} \in N \cup T \cup \{B\}, \mathbf{b} \in N \cup T \cup \{B\} \cup \{\diamond\}, \gamma \in \{\alpha, \beta\}$.
Posons aussi $\mathcal{V} = N \cup T \cup \{B\} \cup \{\alpha, \beta\} \cup \{\diamond\}$.

L'alphabet V est défini par :
$V = \mathcal{V} \cup \{X, Y, X_\alpha, X_\beta, X'_\alpha, Y_\alpha, Y'_\alpha, Y_\beta, Z, Z', Z_j, R_j, L_j\}$.

L'alphabet terminal T est celui de la grammaire formelle G.

Les tubes du système sont définis de la manière suivante.

Tube I :

Règles de R_1 :

$1.1.1 : \dfrac{\varepsilon \mid uY}{R_1 \mid vY}$; $\quad 1.1.2 : \dfrac{X \mid \mathbf{a}}{X_\alpha\beta \mid L_1}$; $\quad 1.1.3 : \dfrac{X \mid \mathbf{a}}{X_\beta\beta\beta \mid L_1}$;

$1.1.4 : \dfrac{\mathbf{a} \mid a_i Y}{R_1 \mid \beta\alpha^{i-1}Y'_\alpha}$; $\quad 1.1.5 : \dfrac{\mathbf{a} \mid BY}{R_1 \mid \diamond Y_\beta}$;

$1.2.1 : \dfrac{\gamma \mid Y'_\alpha}{R_2 \mid Y_\alpha}$; $\quad 1.2.2 : \dfrac{X_\alpha \mid \gamma}{X'_\alpha\alpha \mid L_2}$;

$1.3.1 : \dfrac{X'_\alpha \mid \alpha}{X_\alpha \mid L_3}$; $\quad 1.3.2 : \dfrac{X'_\alpha \mid \alpha}{X_\beta\beta \mid L_3}$; $\quad 1.3.3 : \dfrac{\gamma \mid \alpha Y_\alpha}{R_3 \mid Y'_\alpha}$;

$1.3.4 : \dfrac{\mathbf{b} \mid \beta Y_\alpha}{R_3 \mid Y_\beta}$;

$1.4.1 : \dfrac{a_i \mid Y_\beta}{R_4 \mid Y}$; $\quad 1.4.2 : \dfrac{X_\beta\beta\alpha^i\beta \mid \mathbf{a}}{Xa_i \mid L_4}$; $\quad 1.4.3 : \dfrac{X_\beta\beta\beta \mid \mathbf{a}}{\varepsilon \mid Z'L_4}$;

$1.4.4 : \dfrac{\mathbf{a} \mid \diamond Y_\beta}{Z'L_4 \mid \varepsilon}$;

$$1.1.1' : \frac{Z_1 \mid vY}{R_1 \mid L_1}\ ; \qquad 1.1.2' : \frac{X_\alpha\beta \mid Z_1}{R_1 \mid L_1}\ ; \qquad 1.1.3' : \frac{X_\beta\beta\beta \mid Z_1}{R_1 \mid L_1}\ ;$$

$$1.1.4' : \frac{Z_1 \mid \beta\alpha^i Y'_\alpha}{R_1 \mid L_1}\ ; \qquad 1.1.5' : \frac{Z_1 \mid \diamond Y_\beta}{R_1 \mid L_1}\ ;$$

$$1.2.1' : \frac{Z_2 \mid Y_\alpha}{R_2 \mid L_2}\ ; \qquad 1.2.2' : \frac{X'_\alpha\alpha \mid Z_2}{R_2 \mid L_2}\ ;$$

$$1.3.1' : \frac{X_\alpha \mid Z_3}{R_3 \mid L_3}\ ; \qquad 1.3.2' : \frac{X_\beta\beta \mid Z_3}{R_3 \mid L_3}\ ; \qquad 1.3.3' : \frac{Z_3 \mid Y'_\alpha}{R_3 \mid L_3}\ ;$$

$$1.3.4' : \frac{Z_3 \mid Y_\beta}{R_3 \mid L_3}\ ;$$

$$1.4.1' : \frac{Z_4 \mid Y}{R_4 \mid L_4}\ ; \qquad 1.4.2' : \frac{Xa_i \mid Z_4}{R_4 \mid L_4}\ ; \qquad 1.4.3' : \frac{Z' \mid Z_4}{R_4 \mid L_4}\ ;$$

Filtres :
$F_1^{(1)} = \mathcal{V} \cup \{X_\alpha, Y'_\alpha, R_2, L_2\}$,
$F_1^{(2)} = \mathcal{V} \cup \{X'_\alpha, Y_\alpha, R_3, L_3\}$,
$F_1^{(3)} = \mathcal{V} \cup \{X_\beta, Y_\beta, R_4, L_4\}$,
$F_1^{(4)} = \mathcal{V} \cup \{X, Y, R_1, L_1\}$.

Axiomes de A_1 :
$\{XBSY, X_\alpha\beta Z_1, X_\beta\beta\beta Z_1, Z_1\beta\alpha^i Y'_\alpha, Z \diamond Y_\beta, Z_2 Y_\alpha, X'_\alpha\alpha Z_2,$
$X_\alpha Z_3, X_\beta\beta Z_3, Z_3 Y'_\alpha, ZY_\beta, Z_4 Y, Xa_i Z_4, Z' Z_4, R_1 L_1\} \cup$
$\{ZvY : \exists u \to v \in P\}$.

Tube II :

Aucune règle : $R_2 = \emptyset$.

Filtres :
$F_2^{(j)} = \mathcal{V} \cup \{X, Y, X_\alpha, X_\beta, X'_\alpha, Y_\alpha, Y'_\alpha, Y_\beta, Z', R_j, L_j\}$.

Axiomes de A_2 :
$\{R_2 L_2, R_3 L_3, R_4 L_4\}$

Le système est semblable à celui que nous avons construit à la section précédente. Il contient les même règles et ces règles sont placées dans le premier tube. Les règles sont divisées en quatre sous-ensembles, $1.1.x$ à $1.4.x$, qui ne peuvent pas tous être utilisés en même temps. Pour atteindre ce but, nous utilisons la méthode des molécules assistantes, voir le chapitre 6 et, en particulier, la section 6.4. Nous utilisons les molécules assistantes $R_j L_j$ qui voyagent d'un tube à l'autre. Lorsque la molécule assistante $R_j L_j$ apparaît dans le premier tube, elle active le sous-ensemble j de règles qui peuvent être appliquées ensuite. Plus précisément, la partie inférieure de chaque règle est faite d'une manière spéciale : le site correspond à une molécule assistante qui est produite uniquement si la molécule $R_j L_j$ correspondant au sous-ensemble est présente dans le premier tube. Par exemple, la règle 1.1.3 est utilisable si et seulement si la molécule $X_\beta\beta\beta L_1$ est présente. Mais cette molécule est présente uniquement si le mot $R_1 L_1$ est présent dans le premier tube, et elle est produite en

appliquant la règle 1.1.3′ à l'axiome $X_\beta\beta\beta Z_1$ et à la molécule assistante R_1L_1. À l'étape suivante le mot $X_\beta\beta\beta L_1$ est envoyé dans le deuxième tube et il n'est plus présent dans le premier.

On a donc une imbrication des molécules assistantes et les molécules assistantes du premier niveau qui correspondent aux sites des règles sont créées à l'aide des molécules assistantes du deuxième niveau, R_jL_j, qui voyagent d'un tube à l'autre et leur apparition est dirigée par les filtres alternants. Les filtres agissent aussi comme des sélecteurs pour les molécules correctes et le calcul est fait de la manière suivante : les molécules sont envoyées dans le deuxième composant d'où les molécules correctes sont extraites dans le premier composant où elles sont transformées ensuite. □

Nous observons que les opérations effectués dans le premier et le troisième groupes sont indépendantes et que les molécules utilisées ont une forme différente. Il est donc possible de combiner le premier et le troisième groupes, ainsi que les filtres correspondants 2 et 4. Nous obtenons donc :

Théorème 7.2.3. *Soit $G = (N, T, P, S)$ une grammaire arbitraire. Il existe un système de tubes à essai avec filtres alternants ayant deux tubes et trois filtres, $\Gamma = \left(V, T, \left(A_1, R_1, F_1^{(1)}, \ldots, F_1^{(3)}\right), \left(A_2, R_2, F_2^{(1)}, \ldots, F_2^{(3)}\right)\right)$ qui simule G et pour lequel $L(\Gamma) = L(G)$. De plus, le deuxième composant de Γ ne contient aucune règle, c'est-à-dire que $R_2 = \emptyset$, et aussi $F_2^{(1)} = \cdots = F_2^{(3)}$.*

7.2.3 $\text{TTF}_{2,2}$ avec poubelle

Dans cette section nous présentons un système de tubes à essai avec filtres alternants qui possède deux composants et deux filtres et qui n'a aucune règle dans le deuxième composant. De plus, les deux filtres de chaque composant diffèrent seulement par une lettre. La preuve de ce résultat est aussi différente : nous simulons une machine de Turing à la place d'une grammaire arbitraire.

Nous définissons la fonction de codage ϕ de la manière suivante. Pour chaque configuration w_1qw_2 d'une machine de Turing M, nous définissons l'application de ϕ : $\phi(w_1qw_2) = Xw_1Sqw_2Y$, où X, Y et S sont trois symboles ne faisant pas partie de T.

Nous définissons aussi l'*entrée* d'un système de tubes à essai avec filtres alternants. Une entrée pour le système Γ est un mot w sur l'alphabet de Γ. Le calcul de Γ sur l'entrée w est effectué en ajoutant w aux axiomes du premier tube et en laissant le calcul de Γ se dérouler comme d'habitude.

Lemme 7.2.4. *Soient $M = (Q, T, a_0, s_0, F, \delta)$ une machine Turing et w une entrée. Il existe un système de tubes à essai avec filtres alternants ayant deux composants et deux filtres $\Gamma = \left(V, T_\Gamma, \left(A_1, R_1, F_1^{(1)}, F_1^{(2)}\right), \left(A_2, R_2, F_2^{(1)}, F_2^{(2)}\right)\right)$ qui, en lui donnant comme entrée $\phi(w)$, simule M sur l'entrée w, c'est-à-dire que :*

1. *pour chaque mot w sur lequel M s'arrête dans la configuration w_1qw_2, le système Γ produit un résultat unique $\phi(w_1qw_2)$.*

2. *pour chaque mot w, sur lequel M ne s'arrête pas, le système Γ produit le langage vide.*

Démonstration.

Soient $T = \{a_0, \ldots, a_{m-1}\}$, $Q = \{q_0, \ldots, q_{n-1}\}$, $\mathbf{a} \in T \cup \{X\}$, $\mathbf{b}, \mathbf{d}, \mathbf{e} \in T$, $\mathbf{c} \in T \cup \{Y\}$, $\mathbf{q} \in Q$, où a_0 est le symbole blanc.

Nous pouvons supposer sans perte de généralité que $w_1 w_1$ ne contient aucun symbole a_0 et que M ne possède pas d'instructions stationnaires.

Nous construisons Γ de la manière suivante :

L'alphabet V est donné par :
$V = T \cup Q \cup \{X, Y, S, S', R, L, R', L', R^R, Z_1^R, Z_X, Z_Y, R_1^L, Z_1^L, R_1', Z_1'\}$.

L'alphabet terminal T est donné par :
$T_\Gamma = T \setminus \{a_0\} \cup \{X, Y, S\} \cup \{q : q \in F\}$.

Les composants sont définis de la manière suivante :

Tube I :

Règles de R_1 :

Pour chaque règle $q_i a_k R a_l q_j \in \delta$ nous avons le groupe suivant de 4 règles :

1.1.1.1. $\dfrac{\mathbf{a}Sq_ia_k \mid Y}{Z_Y \mid a_0 Y}$, 1.1.1.2. $\dfrac{\mathbf{a} \mid Sq_ia_k\mathbf{b}}{R \mid L}$,

1.1.1.3. $\dfrac{RSq_ia_k \mid \mathbf{b}}{R_1^R a_l S' q_j \mid Z_1^R}$, 1.1.1.4. $\dfrac{\mathbf{a} \mid L}{R_1^R \mid \mathbf{b}S'\mathbf{qd}}$,

Pour chaque règle $q_i a_k L a_l q_j \in \delta$ nous avons le groupe suivant de 5 règles :

1.1.2.1. $\dfrac{X \mid Sq_ia_k}{Xa_0 \mid Z_X}$, 1.1.2.2. $\dfrac{\mathbf{b} \mid \mathbf{d}Sq_ia_k}{R \mid L}$, 1.1.2.3. $\dfrac{R\mathbf{b}Sq_ia_k \mid \mathbf{c}}{R_1^L S' q_j \mathbf{b} a_l \mid Z_1^L}$,

1.1.2.1'. $\dfrac{X \mid \mathbf{b}Sq_ia_k}{Xa_0 \mid Z_X}$ 1.1.2.4. $\dfrac{\mathbf{b} \mid L}{R_1^L \mid S'\mathbf{qdec}}$,

Nous avons aussi le groupe suivant de 3 règles :

1.2.1. $\dfrac{\mathbf{ab} \mid S'\mathbf{qd}}{R' \mid L'}$, 1.2.2. $\dfrac{R'S' \mid \mathbf{qb}}{R_1'S \mid Z_1'}$, 1.2.3. $\dfrac{\mathbf{b} \mid L'}{R_1' \mid S\mathbf{qd}}$,

Finalement, nous avons le groupe suivant de 3 règles :

1.3.1. $\dfrac{\mathbf{b} \mid a_0 Y}{Z_Y \mid Y}$, 1.3.2. $\dfrac{Xa_0 \mid \mathbf{b}}{X \mid Z_X}$, 1.3.3. $\dfrac{Xa_0 \mid S}{X \mid Z_X}$.

Filtres :
$F_1^{(1)} = \{R', L, L'\}$,
$F_1^{(2)} = \{R, L, L'\} = (F_1^1 \setminus \{R'\}) \cup \{R\}$.

Axiomes de A_1 :
$\{R_Y a_0 Y, X a_0 Z_X, R'_1 S Z'_1, RL\} \cup \{R_1^R a_l S' q_j Z_1^R : \exists q_i a_k R a_l q_j \in \delta\} \cup$
$\{R_1^L S' q_j \mathbf{b} a_l Z_1^L : \exists q_i a_k L a_l q_j \in \delta\}$.

Tube II :

Aucune règle : R_2 est vide.

Filtres :
$F_2^{(1)} = Q \cup T \cup \{X, Y, S, R, L, R', L', R_1^L, R_1^R, R'_1\}$,
$F_2^{(2)} = Q \cup T \cup \{X, Y, S', R, L, R', L', R_1^L, R_1^R, R'_1\} = (F_2^1 - \{S\}) \cup \{S'\}$.

Axiomes de A_2 :
$\{R'L'\}$.

La partie du ruban de M qui contient de l'information est codé de la manière suivante : pour chaque configuration $w_1 q w_2$ de M il existe une configuration $X w_1 S q w_2 Y$ dans Γ. Le ruban est donc entouré par X et Y et nous avons le marqueur Sq_i qui marque sur le ruban la position de la tête de la machine et son état.

Le système Γ simule chaque étape de M en deux étapes. Pendant la première étape il simule une étape de la machine de Turing et il prime le symbole S par les règles $1.1.x.2$ à $1.1.x.4$. Nous disons que ces règles sont associées à la règle correspondante de la machine M. Pendant la seconde étape le système enlève le prime de S' à l'aide des règles $1.2.x$. Le ruban peut être étendu par les règles $1.1.x.1$, ou restreint par les règles $1.3.x$. Le système produit un résultat si et seulement si M s'arrête sur le mot initial w.

Pour effectuer la séparation en deux étapes nous utilisons deux sous-ensembles de règles dans le premier tube, ainsi que les molécules assistantes RL et $R'L'$ qui apparaissent d'une manière alternative dans le premier et le deuxième tube en activant le premier ($1.1.x$) ou le deuxième ($1.2.x$) sous-ensemble de règles, voir aussi la section 7.2.2 et le chapitre 6. Toutes les molécules qui ne sont pas de la forme correcte sont envoyées au deuxième composant qui est considéré comme une poubelle, car seules les molécules assistantes RL et $R'L'$ peuvent passer du deuxième tube au premier.

Les molécules $R_Y a_0 Y$, $R_1^R a_l S' q_j Z_1^R$, $X a_0 Z_X$, $R_1^L S' q_j \mathbf{b} a_l Z_1^L$ et $R'_1 S Z'_1$ sont présentes tout le temps dans le tube 1. De même, toutes les molécules contenant le symbole Z avec des indices restent dans le premier tube.

Nous commençons avec $X w_1 S q_0 w_2 Y$ dans le premier tube, où $w_1 q_0 w_2$ est la configuration initiale de M.

La simulation

Nous allons décrire la simulation de l'application de la règle $q_i a_k R a_l q_j \in \delta$, à la configuration $X w_1 S q_i a_k w_2 \mathrm{Y}$. Nous omettrons de notre description les molécules $R_Y a_0 Y$, $R_1^R a_l S' q_j Z_1^R$, $X a_0 Z_X$, $R_1^L S' q_j \mathbf{b} a_l Z_1^L$ et $R'_1 S Z'_1$ qui sont toujours présentes dans le premier tube. D'une manière similaire, les molécules contenant Z avec des indices et qui n'altèrent pas le calcul seront omises de la description après une étape de calcul.

Nous observons que le deuxième composant peut contenir d'autres molécules, obtenues aux étapes précédentes, car il agit comme une poubelle. Nous omettrons aussi ces molécules, car elles n'altèrent pas notre simulation.

Étape 1.

Recombinaison :

Tube 1 :

Nous avons : $Xw_1Sq_ia_kw_2Y, RL$.

Nous pouvons appliquer les règles suivantes qui font partie du groupe 1.1.1.x :

$(Xw_1|Sq_ia_kw_2Y, R|L) \vdash_{1.1.1.2} (Xw_1L, RSq_ia_kw_2Y)$.

$(RSq_ia_k|w_2Y, R_1^Ra_lS'q_j|Z_1^R) \vdash_{1.1.1.3} (R_1^Ra_lS'q_jw_2Y, RSq_ia_kZ_1^R)$.

$(Xw_1|L, R_1^R|a_lS'q_jw_2Y) \vdash_{1.1.1.4} (Xw_1a_lS'q_jw_2Y, R_1^RL)$.

Tube 2 :

Nous avons : $R'L'$. Comme nous l'avons dit plus haut, nous pouvons avoir ici d'autres molécules obtenues pendant une étape précédente de calcul qui n'altèrent pas le calcul. Nous les omettrons et nous ne mentionnerons plus ce fait dans le futur.

Pas d'application de règles.

Communication :

Les filtres courants sont :

$F_1^{(1)} = \{R', L, L'\}$ et $F_2^{(1)} = Q \cup T \cup \{X, Y, S, R, L, R', L', R_1^L, R_1^R, R_1'\}$.

Molécules envoyées du tube 1 au tube 2 :

$RL, Xw_1Sq_ia_kw_2Y, Xw_1L, RSq_ia_kw_2Y, R_1^RL$.

Molécules restant dans le tube 1 :

$R_1^Ra_lS'q_jw_2Y, RSq_ia_kZ_1^R, Xw_1a_lS'q_jw_2Y$.

Molécules envoyées du tube 2 au tube 1 :

$R'L'$.

Molécules restant dans le tube 2 :

aucune.

La molécule $RSq_ia_kZ_1^R$ ne peut plus évoluer, nous l'omettrons donc dans le futur. De même, la molécule $R_1^Ra_lS'q_jw_2Y$ sera envoyée du premier tube au deuxième tube à l'étape suivante.

Étape 2.

Recombinaison :

Tube 1 :

Nous avons : $R_1^Ra_lS'q_jw_2Y, Xw_1a_lS'q_jw_2Y, R'L'$.

Nous appliquons les règles suivantes qui font partie du groupe 1.2.x :

$(Xw_1a_l|S'q_jw_2Y, R'|L') \vdash_{1.2.1} (Xw_1a_lL', R'S'q_jw_2Y)$.

$(R'S'|q_jw_2Y, R_1'S|Z_1') \vdash_{1.2.2} (R_1'Sq_jw_2Y, R'S'Z_1')$.

$(Xw_1a_l|L', R_1'|Sq_jw_2Y) \vdash_{1.2.3} (Xw_1a_lSq_jw_2Y, R_1'L')$.

Tube 2 :

Nous avons : $Xw_1Sq_ia_kw_2Y, RL, Xw_1L, RSq_ia_kw_2Y, R_1'L'$.

Pas d'application de règles.

Communication :

Les filtres courants sont :

$F_1^{(1)} = \{R', L, L'\}$ et $F_2^{(2)} = Q \cup T \cup \{X, Y, S', R, L, R', L', R_1^L, R_1^R, R_1'\}$.

Molécules envoyées du tube 1 au tube 2 :
$R'L', R_1^R a_l S' q_j w_2 Y, X w_1 a_l S' q_j w_2 Y, X w_1 a_l L', R' S' q_j w_2 Y, R_1' L'.$
Molécules restant dans le tube 1 :
$R' S' Z_1, R_1' S q_j w_2 Y, X w_1 a_l S q_j w_2 Y.$
Molécules envoyées du tube 2 au tube 1 :
$RL.$
Molécules restant dans le tube 2 :
$R'L', R_1^R a_l S' q_j w_2 Y, X w_1 a_l S' q_j w_2 Y, X w_1 a_l L', R' S' q_j w_2 Y, R_1' L',$
$X w_1 S q_i a_k w_2 Y, X w_1 L, RS q_i a_k w_2 Y, R_1^R L.$

De même, la molécule $R_1' S q_j w_2 Y$ sera envoyée du premier tube au deuxième tube à l'étape suivante.

Nous avons donc simulé l'application de la règle $q_i a_k R a_l q_j$ de la machine M. Il est facile d'observer que si $w_2 = \varepsilon$, nous appliquons d'abord la règle 1.1.1.1 pour élargir le ruban à droite et après $w_2 = a_0$. Nous remarquons, que l'application des règles d'élargissement et de restriction du ruban produit plusieurs copies de la molécule principale qui code l'état de la machine de Turing. Ces copies diffèrent par le nombre de symboles a_0 situés à leurs extrémités et elles n'altèrent pas le calcul, même si les règles $1.1.x.4$ s'appliquent aux parties issues de molécules différentes. Nous remarquons aussi qu'il existe une seule molécule qui ne possède aucun symbole a_0 aux extrémités. De même, il est facile de contrôler que les molécules qui ont été produites et qui n'ont pas la forme correcte ($X w_1 S q_i a_k w_2 Y$, $X w_1 L$, $RS q_i a_k w_2 Y$, $R_1^R L$, $R_1^R a_l S' q_j w_2 Y$, $RS q_i a_k Z_1^R$, $X w_1 a_l S' q_j w_2 Y$, $R' S' q_j w_2 Y$, $R' S' Z_1$, $R_1' S q_j w_2 Y$, $X w_1 a_l S' q_j w_2 Y$, $X w_1 a_l L'$) soit sont envoyées au deuxième composant, soit n'altèrent pas le calcul.

Pour le déplacement à gauche, nous procédons de la même manière, sauf que nous nous assurons d'avoir au moins deux symboles à gauche de S.

Nous observons que nous simulons étape par étape l'application des règles de M. Il est facile d'observer que c'est la seule possibilité d'effectuer le calcul, car la molécule qui code la configuration de M déclenche l'application des règles de Γ.

□

Théorème 7.2.5. *Soit $\mathcal{L} \subseteq T^*$ un langage récursivement énumérable. Il existe un système de tubes à essai avec filtres alternants ayant deux composants et deux filtres $\Gamma = \left(V, T, \left(A_1, R_1, F_1^{(1)}, F_1^{(2)}\right), \left(A_2, R_2, F_2^{(1)}, F_2^{(2)}\right)\right)$, qui produit $\mathcal{L}$, c'est-à-dire que $\mathcal{L} = L(\Gamma)$. De plus, $R_2 = \emptyset$.*

Démonstration. Nous procédons d'une manière similaire à ce qui a été fait dans [24] et [26].

Soit $\mathcal{L} = \{w_0, w_1, \dots\}$. Il existe une machine de Turing $T_{\mathcal{L}}$ qui calcule $w_i 0 \dots 0$ à partir de 01^{i+1} où 0 est le symbole blanc.

Il est possible de construire une telle machine de la manière suivante. Soit G une grammaire qui produit $\mathcal{L}$. Nous pouvons construire une machine de Turing M_1 qui simule les dérivations de G, c'est-à-dire que nous obtiendrons toutes les formes syntaxiques de G. Nous devons simuler des dérivations bornées pour prévenir une

récurrence éventuelle dans la dérivation. Après avoir dérivé un nouveau mot, la machine contrôle si ce mot est terminal, et, si c'est le cas, elle contrôle si c'est le $i^{\text{ième}}$ terminal obtenu. Si la dernière condition est vraie, la machine efface tout le ruban, excepté le mot.

En outre, cette machine n'est pas stationnaire, c'est-à-dire que la tête doit se déplacer à chaque étape, et elle ne visite jamais les cases situées à gauche de 0 dans la configuration initiale, c'est-à-dire que le ruban est semi-infini à droite. Dans l'ouvrage [18] on peut trouver comment il est possible de satisfaire ces conditions. La machine $T_{\mathcal{L}}$ transforme donc la configuration $q_0 01^{k+1}$ en $q_f w_k 0 \ldots 0$.

Maintenant, à partir de la machine $T_{\mathcal{L}}$, il est possible de construire une machine $T'_{\mathcal{L}}$ qui calcule $01^{k+2} M q'_f w_k 0 \ldots 0$ à partir de $q_0 01^{k+1}$. Puis, en utilisant Γ nous détachons w_k et nous continuons le calcul à partir de la configuration $q_0 01^{k+2}$. Par ce procédé, nous produisons $\mathcal{L}$ mot par mot.

Pour simplifier la démonstration, nous considérons la machine $T''_{\mathcal{L}}$ qui fait la même chose que la machine $T'_{\mathcal{L}}$, mais à la fin elle déplace sa tête à droite jusqu'au premier zéro, c'est-à-dire qu'elle produira $01^{k+2} M w_k q''_f 0 \ldots 0$ à partir de $q_0 01^{k+1}$. Nous utiliserons aussi deux machines de Turing T_1 et T_2 ayant comme alphabet de ruban l'alphabet de $T''_{\mathcal{L}}$ étendu par l'ensemble $\{X, Y, S\}$. La première machine, T_1, se déplace à gauche jusqu'à ce qu'elle rencontre M, puis elle s'arrête dans l'état q^1_f. L'état initial de T_1 est q^1_s. La deuxième machine, T_2, se déplace à gauche jusqu'à ce qu'elle rencontre 0, puis elle s'arrête dans l'état q_0. L'état initial de cette machine est q^2_s.

Considérons maintenant Γ qui est très similaire au système construit dans le lemme précédent. En effet, nous prenons la machine $T''_{\mathcal{L}}$ et nous construisons Γ pour cette machine comme il a été fait plus haut. Puis, nous prenons les règles de T_1 et T_2 et nous ajoutons les règles de recombinaison associées et les axiomes qui leurs correspondent à Γ. Finalement, nous ajoutons les règles de recombinaison suivantes :

$$x.1.\ \frac{\mathbf{a} \mid \mathbf{bd} S q''_f 0}{Z_1 \mid S q^1_s \mathbf{bd}}, \qquad x.2.\ \frac{\mathbf{e} \mid 11 S q^1_f M}{X_2 \mid \mathbf{e} S q^2_s 11 Y}, \qquad x.2'.\ \frac{X_2 11 S q^1_f M \mid \mathbf{c}}{\varepsilon \mid Z'}$$

où $\mathbf{e} = \{0, 1\}$.

Nous ajoutons aussi $X q_0 01 Y$, ainsi que les mots $Z_1 S q^1_s \mathbf{bd}$, $X_2 \mathbf{e} S q^2_s 11 Y$ et Z' aux axiomes du premier tube et nous considérons T comme alphabet terminal.

Nous affirmons que Γ produit $\mathcal{L}$. Premièrement, ce système simule la machine $T''_{\mathcal{L}}$, il permet donc de passer de la configuration $X S q_0 01^{k+1} Y$ à la configuration $X 01^{k+2} M w_k S q''_f 0 \ldots 0 Y$. Nous pouvons supposer sans perte de généralité que w_k a au moins trois lettres. Maintenant, nous pouvons utiliser la règle $x.1$. Cette règle détache le mot $0 \ldots 0 Y$ qui est envoyé dans le deuxième tube et il nous reste la molécule $X 01^{k+2} M w'_k S q^1_s ab$ ($w_k = w'_k ab$). Nous observons que maintenant nous pouvons simuler le travail de T_1, car la seule condition qui doit être satisfaite est la présence d'un symbole à la droite de la position de la tête de la machine. À partir de ce moment, le système Γ simulera donc le travail de T_1 et nous arriverons à la molécule suivante : $X 01^{k+2} S q^1_f M w_k$. À cet instant, les règles $x.2$ et $x.2'$ détachent

w_k qui représentera le résultat, et nous obtenons la molécule $X01^k Sq_s^2 11Y$. D'une manière similaire, nous pouvons simuler le travail de T_2 et nous arriverons dans la configuration $XSq_0 01^{k+2}Y$. Maintenant nous pouvons recommencer le calcul ci-dessus et produire w_{k+1}, par conséquent nous produirons tous les mots de $\mathcal{L}$.

□

7.3 Conclusions

Nous avons vu dans ce chapitre la puissance de l'élimination. En effet, comme le montre les exemples de ce chapitre, il suffit d'ajouter une possibilité d'élimination aux systèmes de Head étendus pour qu'ils acquièrent une grande puissance de calcul. Les molécules assistantes jouent aussi un rôle important. Nous avons vu qu'en les utilisant il est possible de raffiner le contrôle d'un système de tubes à essai ayant deux tubes à un tel point qu'il n'est plus nécessaire d'avoir de règle dans le deuxième tube.

Le contrôle des systèmes à tubes à essai avec filtres alternants est très puissant et il permet d'appliquer facilement la méthode des molécules assistantes. Dans le chapitre suivant nous montrons un autre exemple d'utilisation de cette méthode pour des systèmes dont le contrôle est beaucoup plus fin.

Chapitre 8

Systèmes de tubes à essai modifiés

Dans ce chapitre nous continuons l'étude des systèmes de tubes à essai et nous introduisons une autre variante de ces systèmes. Nous intervenons comme avant au niveau du protocole de communication, mais cette fois-ci nous n'ajoutons aucun ingrédient supplémentaire. La modification que nous faisons vise les molécules qui peuvent partir d'un tube. Dans ce cas, nous ne permettrons plus à ces molécules de rester dans le tube initial, même si elles peuvent franchir son filtre. Par conséquent, nous pouvons diriger jusqu'à une certaine limite l'élimination de ces molécules. Le contrôle obtenu est beaucoup plus fin que le contrôle par les filtres alternants, mais il nécessite une manipulation délicate. Nous montrons que deux tubes suffisent pour produire tous les langages récursivement énumérables. La méthode des molécules assistantes joue un rôle important dans la démonstration de ce résultat qui est très difficile à obtenir autrement. Une autre particularité intéressante du système obtenu est le processus d'élimination qui n'est pas statique comme dans le cadre des systèmes de tubes à essai avec filtres alternants où on envoie les molécules incorrectes dans l'autre tube où elles ne peuvent plus évoluer, mais dynamique et la molécule éliminée voyage d'un tube à l'autre sans pouvoir participer à une recombinaison, car à l'étape correspondante il n'existe pas de molécule qui lui soit complémentaire par rapport à quelque règle que ce soit.

8.1 Systèmes de tubes à essai modifiés

Définition 8.1.1. Un *système de n tubes à essai modifiés* est la construction suivante :

$$\Gamma = (V, T, (A_1, R_1, F_1), \ldots, (A_n, R_n, F_n)),$$

où V, T, A_i, R_i et F_i sont définis comme dans le cas des systèmes de tubes à essai.

Les systèmes de tubes à essai modifiés et les systèmes décrits au chapitre précédent possèdent un protocole de communication différent. Plus exactement, on définit la transition entre deux configurations de la manière suivante :

$(L_1, \ldots, L_n) \Rightarrow (L'_1, \ldots, L'_n)$ ssi

$$L'_i = \left(\bigcup_{\substack{j=1 \\ j \neq i}}^{n} \sigma_j^*(L_j) \cap F_i^* \right) \cup \left(\sigma_i^*(L_i) \cap \left(V^* - \bigcup_{k=1}^{n} F_k^* \right) \right).$$

On voit que dans l'union de la première parenthèse le cas $j = i$ est exclus.

En termes usuels, ceci signifie que si une molécule peut franchir un filtre d'un tube différent de celui dans lequel elle a été produite, elle ne reste pas dans le tube initial, même si elle peut franchir son filtre. Cette modification permet d'organiser une élimination des molécules et, par conséquent, d'appliquer la méthode des molécules assistantes.

Cette définition est applicable dans le cas où on a au moins deux tubes. Lorsque $n = 1$, nous posons, par définition, que $L'_1 = \sigma_1^*(A_1)$.

Nous notons par TTM_n la famille des langages produits par les systèmes de tubes à essai modifiés ayant au plus n tubes.

Nous remarquons la propriété suivante des systèmes introduits ci-dessus. Soit Γ un système de tubes à essai modifiés ayant 2 tubes. Soit M une molécule qui peut franchir les filtres des deux tubes et supposons qu'initialement elle se trouve dans le tube 1. Il est facile de voir qu'après la première étape de calcul, M est envoyée dans le deuxième tube et aucune copie de M ne reste dans le premier tube. À l'étape suivante, comme M peut franchir le premier filtre, cette molécule revient au premier tube et aucune copie d'elle ne reste dans le deuxième tube. À chaque étape M change donc de tube, c'est-à-dire qu'elle apparaît dans chaque tube avec une période égale à deux. Par ce procédé il est donc possible de créer des molécules assistantes de période 2 et dont l'apparition est dirigée par le contrôle, voir aussi la section 6.4.

Théorème 8.1.1. *Soit $G = (N, T, P, S)$ une grammaire arbitraire. Il existe un système de tubes à essai modifiés ayant deux tubes, $\Gamma = (V, T, (A_1, R_1, F_1), (A_2, R_2, F_2))$ qui simule G et pour lequel on ait $L(\Gamma) = L(G)$.*

Démonstration. Nous construisons Γ de la manière suivante.

Soient $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(B = a_n)$ et $B \notin N \cup T$.

Dans ce qui suit nous supposons que :

$\mathbf{a} \in V \cup T \cup \{B\}$, $\mathbf{b} \in V \cup T \cup \{B\} \cup \{\alpha\}$, $\mathbf{c} \in V_\delta$, $\gamma \in \{\alpha, \beta\}$.

Posons $V_\delta = V \cup T \cup \{B\} \cup \{\alpha, \beta\}$.

Posons aussi $V_{AB} = \{A_x, B_x\}, x \in \{X'_\beta\beta, Y'_\beta, X'_\alpha, Y'_\alpha, X, Y, X_\alpha\alpha, Y_\alpha, X', Y'\}$ et $V_{CD} = \{D_x, C_y\}$, où $x \in \{X, X'_\beta\beta, X'_\alpha, X_\alpha\alpha, X'\}$ et $y \in \{Y'_\alpha, Y, Y'_\beta, Y_\alpha, Y'\}$.

L'alphabet V est défini par :

$V = V_\delta \cup V_{AB} \cup V_{CD} \cup \{Z_X, Z_{BY}, Z_v, Z_{Y_i}, Z_{X_i}\} \cup$
$\{X, Y, X', Y', X_\alpha, Y_\alpha, X_\beta, Y_\beta, X'_\alpha, Y'_\alpha, X'_\beta, Y'_\beta\}$.

Les tubes du système sont définis de la manière suivante.

Tube I :

Axiomes de A_1 :

$\{XBSY\} \cup \{Z_v vY : u \to v \in P\} \cup \{C_{Y'_\alpha} Y'_\alpha, XD_X, C_Y Y, X'_\beta\beta D_{X'_\beta\beta}, C_{Y'_\beta} Y'_\beta, X'_\alpha D_{X'_\alpha}\} \cup$
$\{Z_{Y_i}\beta\alpha^i Y, Xa_i Z_{X_i}, Z_{Y_\beta} Y_\beta\} \cup \{A_{Y'_\alpha} B_{Y'_\alpha}, B_X A_X\} \cup \{B_{X_\alpha} A_{X_\alpha}, A_{Y_\alpha} B_{Y_\alpha}, A_{Y'} B_{Y'}\}$.

Règles de R_1 :

$$1.3.1 : \frac{\varepsilon \mid uY}{Z_v \mid vY} \quad 1.3.2 : \frac{\mathbf{a} \mid a_iY}{Z_{Y_i} \mid \beta\alpha^iY} \quad 1.3.3 : \frac{X\beta\alpha^i \mid \mathbf{a}}{Xa_i \mid Z_{X_i}} \quad 1.3.4 : \frac{\mathbf{a} \mid \beta Y}{Z_{Y_\beta} \mid Y_\beta}$$

$$1.1.1 : \frac{\gamma \mid Y_\alpha}{A_{Y'_\alpha} \mid Y'_\alpha} \qquad 1.1.2 : \frac{X' \mid \gamma}{X \mid A_X}$$

$$1.2.1 : \frac{X \mid \alpha}{X'_\beta\beta \mid A_{X'_\beta\beta}} \quad 1.2.2 : \frac{\mathbf{a} \mid Y_\beta}{A_{Y'_\beta} \mid Y'_\beta} \quad 1.2.3 : \frac{X_\alpha \mid \mathbf{b}}{X'_\alpha \mid A_{X'_\alpha}} \quad 1.2.4 : \frac{\mathbf{c} \mid Y'}{A_Y \mid Y}$$

$$BA.x : \frac{x \mid D_x}{B_x \mid A_x}, \quad \forall x D_x \in A_1, \qquad AB.y : \frac{C_y \mid y}{A_y \mid B_y}, \quad \forall C_y y \in A_1$$

Filtre F_1 :

$V_\delta \cup V_{AB} \cup \{X_\alpha, Y_\alpha, X', Y'\}$.

Tube I :

Axiomes de A_2 :

$\{X_\alpha\alpha D_{X_\alpha\alpha}, C_{Y_\alpha}Y_\alpha, C_{Y'}Y', X'D_{X'}\} \cup \{Z_X, Z_{BY}\} \cup \{B_{X'}A_{X'}\} \cup$
$\{B_{X'_\beta}A_{X'_\beta}, A_{Y'_\beta}B_{Y'_\beta}, B_{X'_\alpha}A_{X'_\alpha}, A_YB_Y\}$.

Règles de R_2 :

$$2.3.1 : \frac{X \mid \mathbf{a}}{\varepsilon \mid Z_X} \qquad 2.3.2 : \frac{\varepsilon \mid BY}{Z_{BY} \mid \varepsilon}$$

$$2.1.1 : \frac{X \mid \mathbf{b}}{X_\alpha\alpha \mid A_{X_\alpha\alpha}} \quad 2.1.2 : \frac{\gamma \mid \alpha Y}{A_{Y_\alpha} \mid Y_\alpha} \quad 2.1.3 : \frac{\gamma \mid Y'_\alpha}{A_{Y'} \mid Y'} \quad 2.1.4 : \frac{\mathbf{a} \mid Y'_\beta}{A_{Y'} \mid Y'}$$

$$2.2.1 : \frac{X'_\alpha \mid \mathbf{b}}{X' \mid A_{X'}} \qquad 2.2.2 : \frac{X'_\beta \mid \beta}{X' \mid A_{X'}}$$

$$BA.x : \frac{x \mid D_x}{B_x \mid A_x}, \quad \forall x D_x \in A_2, \qquad AB.y : \frac{C_y \mid y}{A_y \mid B_y}, \quad \forall C_y y \in A_2$$

Filtre F_2 :

$V_\delta \cup V_{AB} \cup \{X, Y, X'_\beta, Y'_\beta, X'_\alpha, Y'_\alpha, X_\alpha, Y'\}$.

Nous affirmons que $L(\Gamma) = L(G)$.

Nous allons prouver cette affirmation de la manière suivante. Nous montrerons comment il est possible de simuler les dérivations de la grammaire formelle G et, par conséquent, nous obtiendrons que $L(G) \subseteq L(\Gamma)$. Dans le même temps, nous montrerons que toutes les autres possibilités de calcul n'aboutissent pas à un mot terminal, ce qui prouvera l'énoncé.

Notre simulation est fondé sur la méthode «faire-tourner-et-simuler». Nous utilisons les mêmes idées que celles utilisées pendant la démonstration du théorème 7.2.1. Plus précisément, nous faisons tourner le mot Xwa_iY d'une lettre en trois étapes. Premièrement, nous codons a_i par $\beta\alpha^i$ et nous obtenons $Xw\beta\alpha^iY$. Puis, nous transférons les α de l'extrémité droite du mot à son extrémité gauche en obtenant $X\beta\alpha^iwY$. Finalement, nous décodons $\beta\alpha^i$ en a_i et nous obtenons Xa_iwY.

Pour réaliser le procédé ci-dessus nous utilisons la méthode des molécules assistantes, voir chapitre 6. Nous considérons deux types de molécules assistantes. Les molécules assistantes du premier type, qui dirigent l'application des règles et qui sont de la forme xD_x ou $C_y y$, sont crées par imbrication à l'aide des molécules assistantes du deuxième type. Les molécules assistantes du deuxième type qui sont de la forme $A_y B_y$ ou $B_x A_x$, sont crées par le contrôle de la manière suivante. Les filtres de chaque tube permettent le passage de ces molécules et, par conséquent, elles apparaissent dans un tube avec une période égale à deux, voir les réflexions données plus haut. On obtient donc que les deux types de molécules assistantes apparaissent avec une période égale à deux.

En utilisant ces molécules assistantes nous séparons les règles du système en trois groupes :

1. Les règles du premier tube qui sont applicables pendant une étape impaire uniquement et les règles du deuxième tube qui sont applicables pendant une étape paire seulement.
2. Les règles du premier tube qui sont applicables pendant une étape paire uniquement et les règles du deuxième tube qui sont applicables pendant une étape impaire seulement.
3. Les règles des deux tubes qui sont applicables à n'importe quel moment.

On dit que les règles sont activées aux étapes pendant lesquelles elles sont applicables et désactivées pendant les autres étapes.

Les numéros des règles permettent de savoir dans quelle groupe la règle correspondante est placée. Plus précisément, le premier numéro d'une règle indique le tube auquel elle appartient, tandis que le deuxième numéro indique le groupe auquel la règle appartient.

En effet, la séparation des règles en ces groupes ci-dessus permet d'obtenir la propriété suivante : si une règle s'applique a une molécule et si le résultat est ensuite envoyé dans l'autre tube où il est changé par une autre règle et est envoyé à nouveau dans le tube initial, les deux règles qui ont été utilisées font partie du même groupe.

Maintenant, nous montrons avec plus de détails comment on active les règles des groupes (1) et (2). Nous prenons comme exemple la règle 1.2.1 : $\dfrac{X \mid \alpha}{X'_\beta\beta \mid A_{X'_\beta\beta}}$.

Dans Γ les objets suivants existent qui sont associés à cette règle :

Dans le tube I :

L'axiome : $X'_\beta\beta D_{X'_\beta\beta}$.

Les règles :

1.2.1 : $\dfrac{X \mid \alpha}{X_\beta\beta \mid A_{X'_\beta\beta}}$ et $BA.X'_\beta$: $\dfrac{X_\beta\beta \mid D_{X'_\beta\beta}}{B_{X'_\beta\beta} \mid A_{X'_\beta\beta}}$.

Une partie du filtre F_1 : $B_{X'_\beta\beta}, A_{X'_\beta\beta}$.

Dans le tube II :

L'axiome : $B_{X'_\beta\beta} A_{X'_\beta\beta}$.

Aucune règle.

Une partie du filtre F_2 : $X'_\beta, B_{X'_\beta\beta}, A_{X'_\beta\beta}$.

La molécule assistante $B_{X'_\beta\beta}A_{X'_\beta\beta}$ voyage d'un tube à l'autre et elle apparaît dans le premier tube pendant une étape paire seulement. Par conséquent, la molécule assistante du premier niveau $X'_\beta\beta A_{X'_\beta\beta}$ apparaît dans le premier tube pendant les étapes paires uniquement et après son utilisation elle est envoyée tout de suite dans le deuxième tube. Ceci implique que la règle 1.2.1 est utilisable pendant une étape paire uniquement, car seule la molécule assistante $X'_\beta\beta A_{X'_\beta\beta}$ s'accorde avec son site inférieur. On obtient donc que la règle 1.2.1 fait partie du groupe (2).

Toutes les autres règles des groupes (1) et (2) sont définies d'une manière similaire.

Nous observons que les règles du troisième groupe sont utilisables à n'importe quel moment, car leur site inférieur correspond à une molécule qui est déjà présente dans le tube correspondant et qui ne peut pas franchir le filtre de l'autre tube à cause de la présence du symbole Z avec des indices.

Notations

Nous observons que le site inférieur de chaque règle correspond soit à une molécule assistante, soit à une molécule qui est déjà présente dans le tube correspondant. De même, une des molécules résultantes soit contient le symbole Z avec des indices et ne peut s'accorder avec aucune règle, soit est envoyé dans l'autre tube où soit elle reste pour toujours, soit elle voyage à chaque étape entre les deux tubes et n'altère pas le calcul. C'est pourquoi nous omettrons ces molécules et nous écrirons :

$Xwa_iY \underset{1.3.2}{\longrightarrow} Xw\beta\alpha^iY$ à la place de

$(Xw|a_iY, Z_{Y_i}|\beta\alpha^iY) \vdash_{1.3.2} (Xw\beta\alpha^iY, Z_{Y_i}a_iY)$

où par | nous avons souligné les sites de recombinaison. Dans ce qui suit nous marquons les molécules qui peuvent évoluer dans le même tube par ⓗ et nous marquons les molécules qui doivent être envoyées dans le premier tube, respectivement le deuxième, par ①, respectivement ②. Nous écrivons aussi $m\uparrow$ si la molécule m ne peut s'accorder avec aucune règle du tube dans lequel elle se trouve et si, dans le même temps, elle ne peut pas être envoyée dans l'autre tube.

Nous allons montrer l'évolution des mots de la forme Xwa_iY. Nous indiquons les règles de recombinaison qui ont été utilisées, ainsi que les molécules résultantes. Nous remarquons, que dans notre système, grâce au parallélisme, plusieurs molécules de cette forme coexistent avec des formes intermédiaires qui évoluent en parallèle. Ces molécules n'interagissent pas les unes avec les autres et nous nous concentrons sur une seule évolution de ce type.

Rotation

Nous montrons comment faire tourner d'une lettre le mot Xwa_iY se trouvant dans le premier tube.

Étape 1.

Tube 1.

Xwa_iY ② $\underset{1.3.2}{\longrightarrow} Xw\beta\alpha^iY$ ②.

Tube 2.

Pas d'applications de règle.

Étape 2.

Tube 1.

Pas d'applications de règle.

Tube 2.

Xwa_iY ⓗ $\underset{2.3.1}{\longrightarrow} wa_iY$ ↑.

Xwa_iY ⓗ $\underset{2.1.1}{\longrightarrow} X_\alpha\alpha wa_iY$ ↑.

$Xw\beta\alpha^iY$ ⓗ $\underset{2.3.1}{\longrightarrow} w\beta\alpha^iY$ ⓗ $\underset{2.1.2}{\longrightarrow} w\beta\alpha^{i-1}Y_\alpha$ ①.

$Xw\beta\alpha^iY$ ⓗ $\underset{2.1.1}{\longrightarrow} X_\alpha\alpha w\beta\alpha^iY$ ⓗ $\underset{2.1.2}{\longrightarrow} X_\alpha\alpha B\beta\alpha^{i-1}Y_\alpha$ ①.

Les deux dernières règles peuvent être appliquées dans l'ordre inverse, mais cela donne le même résultat.

Étape 3.

Tube 1.

$w\beta\alpha^{i-1}Y_\alpha$ ⓗ $\underset{1.1.1}{\longrightarrow} w\beta\alpha^{i-1}Y'_\alpha$ ②.

$X_\alpha\alpha w\beta\alpha^{i-1}Y_\alpha$ ⓗ $\underset{1.1.1}{\longrightarrow} X_\alpha\alpha w\beta\alpha^{i-1}Y'_\alpha$ ⓗ.

Tube 2.

Pas d'applications de règle.

Étape 4.

Tube 1.

$X_\alpha\alpha w\beta\alpha^{i-1}Y_\alpha$ ⓗ $\underset{1.2.3}{\longrightarrow} X'_\alpha\alpha w\beta\alpha^{i-1}Y_\alpha$ ⓗ

$X_\alpha\alpha w\beta\alpha^{i-1}Y'_\alpha$ ⓗ $\underset{1.2.3}{\longrightarrow} X'_\alpha\alpha w\beta\alpha^{i-1}Y'_\alpha$ ②

Tube 2.

$w\beta\alpha^{i-1}Y'_\alpha$ ⓗ $\underset{2.1.3}{\longrightarrow} w\beta\alpha^{i-1}Y'$ ①

Étape 5.

Tube 1.

$w\beta\alpha^{i-1}Y'$ ②.

$X'_\alpha\alpha w\beta\alpha^{i-1}Y_\alpha$ ⓗ $\underset{1.1.1}{\longrightarrow} X'_\alpha\alpha w\beta\alpha^{i-1}Y'_\alpha$ ②.

Tube 2.

$X'_\alpha\alpha w\beta\alpha^{i-1}Y'_\alpha$ ⓗ $\underset{2.2.1}{\longrightarrow} X'\alpha w\beta\alpha^{i-1}Y'_\alpha$ ⓗ.

Le mot $w\beta\alpha^{i-1}Y'$ n'est pas changé, car on est à une étape impaire et la règle 1.2.4 n'est pas activée. Il voyagera donc d'un composant à l'autre sans être changé, par conséquent il ne pourra pas participer à la production du résultat.

Étape 6.

Tube 1.

Pas d'applications de règle.

Tube 2.

$X'_\alpha\alpha w\beta\alpha^{i-1}Y'_\alpha$ ⓗ $\underset{2.1.3}{\longrightarrow} X'_\alpha\alpha w\beta\alpha^{i-1}Y'$ ⓗ.

$X'\alpha w\beta\alpha^{i-1}Y'_\alpha$ ⓗ $\underset{2.1.3}{\longrightarrow} X'\alpha w\beta\alpha^{i-1}Y'$ ①.

Étape 7.

Tube 1.

$X'\alpha w\beta\alpha^{i-1}Y'$ ⓗ $\underset{1.1.2}{\longrightarrow} X\alpha w\beta\alpha^{i-1}Y'$ ②.

Tube 2.

$X'_\alpha\alpha w\beta\alpha^{i-1}Y'$ ⓗ $\underset{2.2.1}{\longrightarrow} X'\alpha w\beta\alpha^{i-1}Y'$ ①.

Étape 8.

Tube 1.

$X'\alpha w\beta\alpha^{i-1}Y'$ ⓗ $\underset{1.2.4}{\longrightarrow} X'\alpha w\beta\alpha^{i-1}Y$ ⓗ.

Tube 2.

$X\alpha w\beta\alpha^{i-1}Y'$ ⓗ $\underset{2.1.1}{\longrightarrow} X_\alpha\alpha\alpha w\beta\alpha^{i-1}Y'$ ①.

Si $i = 1$ nous pouvons appliquer aussi la règle 1.3.4, mais cette application sera discutée plus tard à l'étape 8i. De même, nous discuterons plus tard l'évolution des mots de la forme $X_\alpha wY'$.

Étape 9.

Tube 1.

$X_\alpha\alpha\alpha w\beta\alpha^{i-1}Y'$ ②.

$X'\alpha w\beta\alpha^{i-1}Y$ ⓗ $\underset{1.1.2}{\longrightarrow} X\alpha w\beta\alpha^{i-1}Y$ ②.

Tube 2.

Pas d'applications de règle.

Nous avons donc fait tourner le mot $Xw\beta\alpha^{i-1}\alpha Y$ d'un α.

Nous continuons jusqu'à ce que nous obtenions $X\alpha^i w\beta Y$ à l'étape 8i+1. Dans ce cas nous avons le calcul suivant que nous détaillons à partir de l'étape précédente, 8i, pour montrer l'application de la règle 1.3.4.

Étape 8i.

Tube 1.

$X'\alpha^i w\beta Y$ ⓗ $\underset{1.3.4}{\longrightarrow} X'\alpha^i wY_\beta$ ⓗ.

Tube 2.

Pas d'applications de règle.

Étape 8i+1.

Tube 1.

$X'\alpha^i wY_\beta$ ⓗ $\underset{1.1.2}{\longrightarrow} X\alpha^i wY_\beta$ ②.

$X'\alpha^i w\beta Y \underset{1.1.2}{\longrightarrow} X\alpha^i w\beta Y \underset{1.3.4}{\longrightarrow} X\alpha^i wY_\beta$ ↑.

Tube 2.

Pas d'applications de règle.

Étape 8i+2.

Tube 1.

$X\alpha^i wY_\beta$ ⓗ $\underset{1.2.2}{\longrightarrow} X\alpha^i wY'_\beta$ ②.

$X\alpha^i wY'_\beta$ ② $\underset{1.2.1}{\longrightarrow} X'_\beta\beta\alpha^i wY'_\beta$ ②.

Les deux applications précédentes peuvent être effectuées dans l'ordre inverse ce qui conduit au même résultat.

Tube 2.

Pas d'applications de règle.

Étape 8i+3.

Tube 1.

Pas d'applications de règle.

Tube 2.

$X'_\beta\beta\alpha^i wY'_\beta$ ⓗ $\underset{2.2.2}{\longrightarrow} X'\beta\alpha^i wY'_\beta$ ⓗ.

Étape 8i+4.

Tube 1.

Pas d'applications de règle.

Tube 2.

$X\beta\alpha^i wY'_\beta$ ⓗ $\underset{2.1.1}{\longrightarrow} X_\alpha\beta\alpha^i wY'_\beta$ ⓗ $\underset{2.1.4}{\longrightarrow} X_\alpha\alpha\beta\alpha^i wY'$ ①.

$X'_\beta\beta\alpha^i wY'_\beta$ ⓗ $\underset{2.1.4}{\longrightarrow} X'_\beta\beta\alpha^i wY'$ ⓗ.

$X'\beta\alpha^i wY'_\beta$ ⓗ $\underset{2.1.4}{\longrightarrow} X'\beta\alpha^i wY'$ ①.

Après cet instant l'évolution est similaire à l'évolution après l'étape 6 et après plusieurs étapes on obtient $X\beta\alpha^i wY$ dans le premier tube. Puis nous pouvons appliquer la règle 1.3.3 : $X\beta\alpha^i wY \underset{1.3.3}{\longrightarrow} Xa_i wY$, par conséquent, nous avons fait tourner Xwa_iY d'une lettre.

Nous observons que malgré la justesse du calcul ci-dessus nous ne pouvons pas être sûrs que le système fonctionne correctement. En effet, le calcul du système dépend de la parité de l'étape de calcul et si une molécule pouvait apparaître pendant une étape avec une autre parité, elle provoquerait un calcul erroné. C'est pourquoi nous montrons ci-dessous que nous ne pouvons pas obtenir des molécules incorrectes pendant une autre étape de calcul que celle montrée ci-dessus.

Pour cela nous analysons les molécules qui peuvent passer du deuxième tube au premier tube et qui sont entourées par des parenthèses X et Y avec des indices. Cela nous donne 4 possibilités :

a) Molécules de type $X_\alpha wY'$.

b) Molécules de type $X'wY_\alpha$.

c) Molécules de type $X_\alpha wY_\alpha$.

d) Molécules de type $X'wY'$.

La première éventualité est possible pendant une étape paire uniquement, car il faut que nous ayons utilisé une des règles 2.1.1, 2.1.3 ou 2.1.4 qui sont du premier groupe. Dans ce cas la molécule $X_\alpha wY'$ apparaît dans le premier tube pendant une étape impaire et les règles avec lesquelles elle s'accorde ne sont pas activées pendant cette étape. Par conséquent, cette molécule est envoyée inchangée dans le deuxième tube. De même, dans ce tube il n'y a pas de règle avec laquelle elle pourrait s'accorder. Comme il a été décrit plus haut, cette molécule changera donc de tube à chaque étape. Par ce procédé nous avons implanté une poubelle qui n'est pas fixe et dont le contenu se déplace dans l'autre composant à chaque étape.

Nous allons montrer que (b) n'est pas possible. Premièrement, nous observons que pour obtenir $X'wY_\alpha$ nous aurions du utiliser les règles 2.2.2 et 2.1.2, par conséquent, nous aurions du avoir $X'_\beta wY$ dans le deuxième tube. Ceci est possible uniquement si ce mot avait été envoyé précédemment depuis le premier tube. Pour avoir le

mot $X'_\beta wY$ dans le premier tube nous aurions du avoir le mot XwY dans le premier tube pendant une étape paire. Il est facile de voir que ce n'est pas possible. Si nous avons le mot XwY dans le premier tube pendant une étape impaire, il est envoyé tout de suite au deuxième tube. De même, si nous avions eu $X'wY'$ et si nous avions appliqué la règle 1.1.2, le mot résultant XwY' aurait tout de suite été envoyé dans le deuxième tube.

Nous analysons maintenant les possibilités restantes. Nous remarquons d'abord que dans le deuxième tube nous pouvons ajouter au plus un α au début d'un mot et effacer au plus un α à la fin. De même, pour obtenir le cas (c) ou (d) il faut appliquer les deux règles une fois, ce qui laisse inchangé le nombre d'α dans le mot.

Si nous avons le cas (c), seuls les mots $X'_\alpha wY'_\alpha$ et $X_\alpha wY'_\alpha$ sont envoyés dans le deuxième tube. Le premier mot représente une évolution correcte et il est transformé en $X'wY'$ ce qui constitue le cas (d). Le deuxième mot peut devenir $X_\alpha wY'$ ce qui représente le cas (a) déjà étudié.

Si nous avons le cas (d), nous obtenons $X'wY'$ dans le premier tube.

Maintenant, si nous sommes pendant une étape impaire, nous appliquons la règle 1.1.2 et éventuellement la règle 1.3.3 en obtenant XwY' qui est envoyé dans le deuxième tube où il peut évoluer en $X_\alpha \alpha wY'$ ce qui constitue le cas (a).

Si nous sommes pendant une étape paire, nous appliquons la règle 1.2.4 en obtenant $X'wY$. Puis, pendant une étape impaire nous appliquons la règle 1.1.2 en obtenant XwY qui représente une évolution correcte. Nous pouvons appliquer aussi les règles 1.3.4, 1.2.2 ou 1.2.1 en obtenant $X'wY_\beta$, XwY_β, XwY'_β ou $X'_\beta wY'_\beta$. Le quatrième mot représente une évolution correcte. Les deux premiers mots peuvent évoluer vers le troisième mot uniquement. Le troisième mot est envoyé dans le deuxième tube où il est transformé en $X_\alpha wY'$ ce qui représente le cas (a).

Simulation des productions et le résultat

Il est facile de voir que nous effectuons une simulation correcte des productions de la grammaire par la règle 1.3.1 et que nous obtenons un mot correct de la grammaire en utilisant les règles 2.3.1 et 2.3.2.

□

8.2 Conclusions

Dans ce chapitre nous finissons l'étude de la méthode des molécules assistantes. Au cours des trois derniers chapitres nous avons montré différents exemples utilisant cette méthode qui permet de simplifier beaucoup les démonstrations. La clé de cette méthode, la création dirigée fondée sur l'élimination, peut être réalisée de différentes manières, même si on n'a pas d'élimination physique. Les systèmes étudiés dans ce chapitre sont intéressants par la façon dont on effectue l'élimination qui est dynamique. De même, la « poubelle » contenant les molécules indésirables voyage d'un composant à l'autre. Ce procédé nécessite une bonne synchronisation des différentes

parties du calcul et peut servir comme point de départ pour une amélioration de la méthode des molécules assistantes.

Nous nous arrêtons donc ici et nous considérons au chapitre suivant des systèmes qui ont une inspiration et une structure différentes, mais qui présentent des liens forts avec des systèmes que nous avons déjà étudiés.

Chapitre 9

Systèmes à membranes avec recombinaison

Aux chapitres précédents nous avons parlé des modèles de calcul inspirés par la façon dont les êtres vivants manipulent leur ADN. Par conséquent, notre point de vue était placé au niveau moléculaire. Dans ce chapitre, nous changeons d'optique et nous nous plaçons au niveau cellulaire en considérant les systèmes à membranes qui sont inspirés des différents processus intra- et inter-cellulaires. La cellule est considérée comme un ensemble de compartiments imbriqués les uns dans les autres et qui contiennent des objets et des règles d'évolution. Le modèle de base ne spécifie ni la nature des objets ni la nature des règles. De nombreuses variantes spécifient ces deux paramètres en obtenant une multitude de modèles de calcul. Nous nous intéressons aux modèles dont les objets sont des chaînes de caractères et dont les règles sont des règles de recombinaison. Les systèmes ainsi obtenus portent le nom de systèmes à membranes avec recombinaison. Il est connu que deux membranes suffisent pour produire tous les langages récursivement énumérables. Nous nous sommes intéressés aux systèmes n'ayant qu'une seule membrane. Nous avons trouvé certaines lacunes dans la définition originelle donnée par Gh. Păun et nous proposons plusieurs variantes pour compléter cette définition. Nous montrons que le choix de la variante est crucial et la puissance d'expression en dépend directement. Nous montrons aussi des similitudes entre les systèmes à membranes avec recombinaison ayant une seule membrane et les systèmes distribués à changement de phase de degré 1. Ces similitudes permettent la simulation de certains types de ces derniers systèmes par les premiers.

Nous étudierons aussi les systèmes à membranes avec recombinaison non-étendus, c'est-à-dire qui n'ont pas d'alphabet terminal. Nous donnons la frontière entre la décidabilité et l'indécidabilité pour ces systèmes en montrant que deux membranes suffisent pour produire tous les langages récursivement énumérables, tandis qu'avec une seule membrane la puissance d'expression est bornée par la famille des langages rationnels.

Dans le reste du chapitre nous nous intéressons à une autre variante des systèmes à membranes avec recombinaison : les systèmes à membranes avec recombinaison et

communication immédiate. Dans ces systèmes, le résultat de l'application des règles de recombinaison est envoyé dans toutes les membranes adjacentes. Ces systèmes possèdent des liens forts avec les systèmes distribués à changement de phase et nous montrons que deux membranes suffisent pour produire tous les langages récursivement énumérables. Dans le même temps, nous montrons qu'avec une membrane on ne peut produire que la famille des langages finis, on obtient donc dans ce cas une frontière entre la décidabilité et l'indécidabilité.

La méthode des molécules assistantes est aussi utilisable pour les systèmes à membranes avec recombinaison. Même si nous ne donnons pas des preuves en utilisant cette méthode, il est toujours possible de l'appliquer et nous l'indiquons par la suite.

9.1 Systèmes à membranes avec recombinaison

Dans cette section nous donnons quelques notions sur les systèmes à membranes et nous introduisons les systèmes à membranes avec recombinaison.

Une *structure de membranes* d'un système à membranes est l'arrangement hiérarchique des membranes imbriquées dans la membrane *principale* qui sépare le système de l'*environnement*. Une membrane qui ne contient aucune membrane est appelée *élémentaire*. Chaque membrane détermine une *région*. La région d'une membrane élémentaire est l'espace contenu dans cette membrane, tandis que la région d'une membrane non-élémentaire est l'espace entre la membrane et les membranes incluses dans celle-ci. La figure 9.1 illustre ces notions. Nous marquons les membranes par des numéros entiers positifs. Nous observons qu'il existe une correspondance bijective entre les membranes et les régions qu'elles déterminent, c'est pourquoi nous ne ferons plus la différence entre les membranes, leurs régions et leurs numéros.

L'arrangement spatial des membranes peut être représenté par une diagramme de Venn, voir figure 9.1, ou par un arbre dont les noeuds représentent les membranes et dont les arêtes sont définies par la relation « être contenu dans », voir figure 9.2. La même structure peut être définie à l'aide d'un mot du langage algébrique où les parenthèses correspondantes ont la même étiquette. Par exemple, la structure représentée à la figure 9.1 peut être décrite par le mot : $[_1\,[_2]_2\,[_3]_3\,[_4\,[_5]_5\,[_6]_6\,]_4\,]_1$.

Définition 9.1.1. Un *système à membranes avec recombinaison* de degré $m \geq 1$ est le $2m+3$-uplet suivant :

$$\Pi = (V, T, \mu, A_1, \ldots, A_m, R_1, \ldots, R_m),$$

où V est un alphabet, $T \subseteq V$ est l'alphabet terminal, μ est une structure de membranes contenant m membranes, $A_i \subseteq V^*$ sont des langages associés aux régions $1, \ldots, m$ de μ, R_i sont des ensembles finis de règles d'évolution de la forme $(r; tar_1, tar_2)$, où r est une règle de recombinaison : $r = u_1 \# u_2 \$ u_3 \# u_4$ et $tar_1, tar_2 \in \{here, in, out\}$ sont des indicateurs de cible.

Nous pouvons aussi écrire $\dfrac{u_1 \mid u_2}{u_3 \mid u_4}\,\dfrac{(tar_1)}{(tar_2)}$ à la place de $(u_1 \# u_2 \$ u_3 \# u_4; tar_1, tar_2)$.

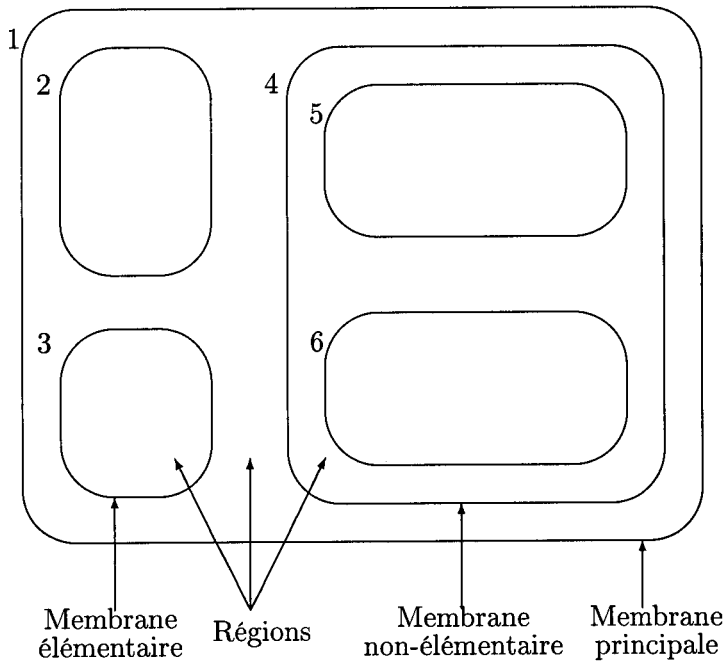


FIG. 9.1 – Une structure de membranes

Nous disons que le système Π possède des règles *globales* si $R_1 = \ldots = R_m$.

Une *configuration* de Π est un m-uplet $(N_1, \ldots, N_m)$, où $N_i \subseteq V^*$. Nous définissons la transition entre deux configurations $(N_1, \ldots, N_m) \Rightarrow (N'_1, \ldots, N'_m)$ de la manière suivante. Pour arriver d'une configuration à l'autre, nous appliquons les règles de recombinaison de chaque région de μ, en parallèle, à tous les mots possibles qui se trouvent dans les régions correspondantes et nous distribuons le résultat de chaque recombinaison en fonction des indicateurs de cible. Plus précisément, s'il existe des x, y dans N_i et $r = (u_1 \# u_2 \$ u_3 \# u_4; tar_1; tar_2)$ dans R_i, tels que $(x, y) \vdash_r (w, z)$, les mots w et z sont envoyés dans les régions indiquées par tar_1, respectivement tar_2. Nous notons ce fait par $(x, y) \vdash_r (w, z)(tar_1, tar_2)$. Si $tar_j = here$, la molécule reste dans la membrane i ; si $tar_j = out$, la molécule est envoyée dans la région située immédiatement à l'extérieur de la membrane i, dans ce cas la molécule peut aussi sortir à l'extérieur du système ; si $tar_j = in$, la molécule est envoyée dans une région située directement à l'intérieur de la membrane i et qui est choisie aléatoirement.

Comme les mots figurent avec un nombre arbitraire de copies, après l'application de la règle r dans la membrane i, les molécules x et y sont toujours présentes dans la même région. Il y a cependant quelques cas particuliers qui sont discutés plus bas.

Un *calcul* dans un système à membranes avec recombinaison Π est une suite de transitions entre les configurations de Π qui commence à partir de la configuration

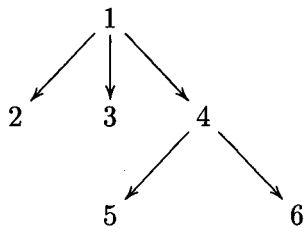


FIG. 9.2 – L'arbre représentant la structure des membranes de la figure 9.1

initiale $(A_1, \ldots, A_m)$. Le résultat du calcul consiste en tous les mots sur l'alphabet terminal T qui sont envoyés à l'extérieur du système à n'importe quel moment du calcul. Nous notons par $L(\Pi)$ le langage produit par le système Π.

Si nous avons une seule membrane, nous omettons les parenthèses et nous écrivons $M_1 \Rightarrow M_2 \Rightarrow \ldots \Rightarrow M_k$ $(M_1 = A_1)$ pour une suite de transitions.

Nous pouvons rencontrer le problème suivant : comment procéder si nous avons un mot w dans une membrane, que nous produisons le même mot w par une certaine règle r et qu'il est envoyé à l'extérieur de la membrane. Il existe deux solutions possibles : la première solution est de garder l'ancien w dans la membrane et d'envoyer une nouvelle copie de w à l'extérieur ; la deuxième solution est d'envoyer w à l'extérieur et de l'éliminer de la membrane.

Un problème semblable se pose dans le cas où il existe deux règles qui produisent le même mot w, l'une ayant l'indicateur de cible *here* et l'autre ayant l'indicateur de cible *in* ou *out*. Les deux situations décrites ci-dessus peuvent apparaître simultanément.

Pour résoudre les problèmes ci-dessus, nous devons considérer tous les cas possibles et nous obtenons plusieurs solutions que nous allons présenter.

Soit maintenant w une molécule qui est produite par une règle r dans la membrane i et qui est envoyée à l'extérieur de la membrane. Nous avons les possibilités suivantes :

1. La molécule w est envoyée à l'extérieur de la membrane et aucune copie de w ne reste dans la même membrane.
2. La molécule w est envoyée à l'extérieur de la membrane, mais une copie de w reste dans la même membrane si l'une des conditions suivantes est satisfaite.
   (a) La molécule w est déjà présente dans la même membrane.
   (b) Il existe une autre règle r' qui produit w avec un indicateur de cible *here*.
   (c) Les deux conditions 2a et 2b sont satisfaites.
   (d) L'une des conditions 2a ou 2b est satisfaite.

Dans la variante 1 on envoie donc tous les occurrences du mot w à l'extérieur de la membrane. Dans les variantes 2a et 2d, si w est produit dans une membrane, il y reste pour toujours. Les variantes 2b et 2c se trouvent entre les deux extrêmes

ci-dessus. Les variantes 2a et 2d ont été considérées par Gh. Păun dans [43], mais sans les différencier.

Nous disons que le système à membranes avec recombinaison est de *type* x si nous considérons la variante x pour résoudre les problèmes décrits plus haut.

Nous notons par $ELSP_m(spl, in)$ la famille des langages produits par les systèmes à membranes de type 2d et de degré au plus m et par $ELSP_m(spl, in_w)$ la famille des langages produits par les systèmes à membranes de type 2a et de degré au plus m. Nous notons aussi par $ELSP_m(spl, in\uparrow)$ la famille des langages produits par les systèmes à membranes de type 1 et de degré au plus m.

Les problèmes ci-dessus peuvent être évités lorsqu'on considère des systèmes ayant plus d'une membrane, voir [36, 9, 43], plus précisément, il est possible de construire un système à membranes avec recombinaison ayant 2 membranes qui produit tous les langages récursivement énumérables sans rencontrer une des situations ci-dessus. Par contre, nous montrerons que dans le cas d'une seule membrane le choix de la définition est très important et les systèmes obtenus ont une puissance d'expression différente. Dans le reste de cette section nous considérons donc uniquement les systèmes à membranes avec recombinaison ayant une seule membrane.

9.1.1 Résultats de décidabilité

Nous montrons ci-dessus les limites de la famille $ELSP_1(spl, in)$.

Considérons les systèmes à membranes avec recombinaison de type 2d. Soit $\Pi = (V, T, [_1]_1, M_1, R_1)$ un tel système. Nous considérons aussi dans cette section la recombinaison simple à la place de la recombinaison double. Cette considération ne restreint pas le langage produit, car nous pouvons remplacer chaque règle de la forme $(u_1\#u_2\$u_3\#u_4; tar_1, tar_2)$ par les règles $(u_1\#u_2\$u_3\#u_4; tar_1)$ et $(u_3\#u_4\$u_1\#u_2; tar_2)$, voir aussi la proposition 3.1.2 à la page 28.

Dans ce cas nous pouvons décomposer R_1 : $R_1 = R_1^h \cup R_1^o$, où R_1^h contient toutes les règles ayant l'indicateur de cible *here* et R_1^o contient toutes les règles ayant l'indicateur de cible *out*. Nous notons aussi par $\sigma_{h_o} = (V \cup T, R_1^o)$ et $\sigma_{h_h} = (V \cup T, R_1^h)$ les schémas de recombinaison correspondants et par $H_h = (h_h, M_1)$ le système de Head correspondant.

Lemme 9.1.1. *$M_{k+1} = \sigma_{h_h}^k(M_1)$, où $M_1 \overset{k}{\Rightarrow} M_{k+1}$.*

Démonstration. Nous allons prouver cette affirmation par récurrence. Le cas initial de la récurrence est vrai, car les deux ensembles sont égaux à M_1. Il est facile de voir que le passage de k à $k+1$ est vrai aussi, car pour obtenir M_{k+1} nous ajoutons à M_k les molécules obtenues par l'application des règles ayant l'indicateur de cible *here* sur l'ensemble M_k, c'est-à-dire que $M_{k+1} = M_k \cup \sigma_{h_h}(M_k)$. □

Lemme 9.1.2. *$L(\Pi) = \sigma_{h_o}(L(H_h)) \cap T^*$.*

Démonstration. Soit $w \in L(\Pi)$. Cela signifie que $w \in T^*$, $w = x_1u_1u_4y_2$ et il existe un nombre $k \in \mathbb{N}$ et des mots $x, y \in M_{k+1}$ tels que $x = x_1u_1u_2x_2$ et $y = y_1u_3u_4y_2$. Il existe aussi une règle $r = (u_1\#u_2\$u_3\#u_4; out) \in R_1^o$ telle que le mot w a été

produit par l'application suivante : $(x, y) \vdash_r w$. En vu du lemme précédent on obtient $x, y \in \sigma^k_{h_h}(M_1) \subseteq L(H_h)$. Par conséquent, $w \in \sigma_{h_o}(L(H_h)) \cap T^*$.

Inversement, si nous avons $w \in \sigma_{h_o}(L(H_h)) \cap T^*$, il existe un nombre $k \in \mathbb{N}$, des mots $x, y \in \sigma^k_{h_h}(M_1)$ et une règle $r \in R_1^o$ tels que $(x, y) \vdash_r w$. Dans ce cas $x, y \in M_{k+1}$, voir lemme 9.1.2, et w est envoyé à l'extérieur du système par r. De même, $w \in T^*$. Cela implique $w \in L(\Pi)$. □

Comme la famille des langages rationnels est close par rapport à la recombinaison et à l'intersection, on obtient le résultat suivant.

Corollaire 9.1.3. $ELSP_1(spl, in) \subseteq REG$.

L'inclusion inverse est aussi vraie.

Lemme 9.1.4. $REG \subseteq ELSP_1(spl, in)$.

Démonstration. Il est facile de voir que nous pouvons simuler un système de Head étendu S par un système à membranes avec recombinaison de type 2d Π qui a une seule membrane de la manière suivante : pour chaque règle r du S nous considérons deux règles $(r; here, here)$ et $(r; out, out)$ dans Π. Dans ce cas, la première règle permet de simuler la règle r du système S, tandis que la deuxième règle permet d'envoyer à l'extérieur de la membrane le résultat du calcul. □

Nous avons donc montré :

Théorème 9.1.5. $ELSP_1(spl, in) = REG$.

9.1.2 Résultats d'indécidabilité

Dans cette section nous considérons la famille $ELSP_m(spl, in \uparrow)$.

Nous allons montrer le théorème suivant.

Théorème 9.1.6. $ELSP_1(spl, in \uparrow) = RE$.

Définissons :

$$Out(r) = \left\{ \frac{u_1 \mid u_2}{u_1 \mid u_2}\ (out),\ \frac{v_1 \mid v_2}{v_1 \mid v_2}\ (out) : r = \frac{u_1 \mid u_2}{v_1 \mid v_2} \right\}.$$

Nous allons montrer maintenant comment il est possible de simuler un système distribué à changement de phase de degré 1 par un système à membranes avec recombinaison de type 1. L'idée principale est de simuler une application de la règle r d'un système distribué à changement de phase de degré 1 par les règles suivantes :

1. $(r; here, here)$.
2. $Out(r)$.

La règle (1) permet de simuler l'application de r, tandis que les règles (2) éliminent les molécules initiales.

Pour démontrer le théorème nous montrons d'abord le lemme suivant.

Lemme 9.1.7. *Soit $D = (V, T, A, R)$ un système distribué à changement de phase de degré 1 qui a les propriétés suivantes :*

- *$L_k \cap L_{k+1} = \emptyset$, c'est-à-dire que les molécules produites à l'étape k et $k+1$ sont différentes.*
- *Pour obtenir un mot-résultat nous utilisons une seule fois une règle de l'ensemble $R_R \subseteq R$ et le deuxième résultat de cette recombinaison peut être éliminé sans altérer le langage produit.*

Alors, il existe un système à membranes avec recombinaison $\Pi \in ELSP_1(spl, in \uparrow)$ qui simule D et pour lequel $L(\Pi) = L(D)$.

Démonstration.
Nous construisons le système à membranes suivant $\Pi = (V, T, [_1]_1, M_1, R')$, où $M_1 = A$ et $R' = R_1 \cup R_2 \cup R_3$ avec

$$R_1 = \{(r, here, here), Out(r) : r \in R \setminus R_R\},$$

$$R_2 = \{(r, out, out), Out(r) : r \in R_R\},$$

$$R_3 = \left\{ \begin{array}{c|c} w & \varepsilon \\ \hline w & \varepsilon \end{array} \begin{array}{c} (out) \\ (out) \end{array} : w \in A \right\}.$$

Nous observons que, par la construction du système Π, si M_k contient une molécule qui ne peut s'accorder avec aucune règle de Π, cette molécule restera pour toujours dans la membrane et ne pourra plus intervenir dans le calcul et produire un mot-résultat. Nous allons considérer la classe d'équivalence définie par la relation « s'accorder avec une règle », c'est-à-dire nous ne considérerons plus les molécules qui ne peuvent s'accorder avec aucune règle de la membrane dans laquelle elles se situent. À chaque étape, toutes les molécules du système s'accordent donc avec une règle de Π et, à cause de la forme des règles, participent à une recombinaison.

Nous affirmons que $L(\Pi) = L(D)$.

Pour prouver cette affirmation, nous devons démontrer les propriétés suivantes :

1. $L_k \setminus T^* = M_k$.
2. $L_{k+1} \cap T^* = \sigma_{h_o}(M_k) \cap T^*$, où $\sigma_{h_o} = (V, R_o)$ et R_o contient toutes les règles de R' ayant l'indicateur de cible *out*.

Démonstration de 1. Nous allons montrer que $L_k \setminus T^* = M_k$, c'est-à-dire que l'ensemble des molécules non-terminales de L_k coïncide avec M_k.

Nous allons prouver cette affirmation par récurrence. Le cas initial de la récurrence est vrai, car les deux ensembles sont égaux à A. Supposons maintenant que nous avons $L_k \setminus T^* = M_k$ et prouvons que $L_{k+1} \setminus T^* = M_{k+1}$.

Soit $w \in L_{k+1} \setminus T^*$. Il existe alors des mots x et y dans L_k et une règle r dans $R \setminus R_R$ tels que w est produit par l'application suivante : $(x, y) \vdash_r (w, z)$. Nous remarquons que la deuxième propriété de D implique que la règle r appartient à l'ensemble $R \setminus R_R$, car, dans le cas contraire, w aurait fait partie de l'alphabet terminal T. Les mots w et z ne sont donc pas des mots terminaux et ils sont différents de x et y grâce à la première propriété de D. Par hypothèse de récurrence, x et y font partie de M_k. De même, par construction, il existe une règle $(r; here, here)$

dans Π que nous pouvons appliquer à x et y en produisant w et z : $(x, y) \vdash_r (w, z)(here, here)$. Nous observons que, grâce à la première propriété de D, les mots w et z sont différents de n'importe quels x et y complémentaires par rapport à une règle de D, c'est-à-dire que nous ne pouvons pas produire les molécules w et z avec l'indication de cible *out*. Les seules molécules qui peuvent donc être envoyées à l'extérieur sont les molécules de M_k, excepté les axiomes qui sont envoyé à l'extérieur pas les règles de R_2. On obtient donc $w, z \in M_{k+1}$.

Inversement, soit $w \in M_{k+1}$. Il existe alors des mots x et y dans M_k complémentaires par rapport à la règle $(r, here, here) \in R'$ tels que w soit produit par l'application suivante : $(x, y) \vdash_r (w, z)(here, here)$. Nous remarquons que si on considère une règle ayant les indicateurs de cible *out*, le résultat de l'application de cette règle ne fera pas partie de M_{k+1}. Les deux résultats w et z restent donc dans la membrane. Dans le même temps, les mots x et y sont envoyés à l'extérieur par les règles de $Out(r)$. Comme x et y sont dans L_k par hypothèse de récurrence et il existe, par construction, une règle $r \in R$, nous avons l'application suivante dans D : $(x, y) \vdash_r (w, z)$, c'est-à-dire que $w, z \in L_{k+1}$.

Nous avons donc montré que $L_{k+1} \setminus T^* = M_{k+1}$.

Démonstration de 2. Nous allons montrer que $L_{k+1} \cap T^* = \sigma_{h_o}(M_k) \cap T^*$, c'est-à-dire que les mots terminaux de L_{k+1} sont les mots terminaux envoyés à l'extérieur de la membrane à l'étape $k + 1$.

Soit $w \in L_{k+1} \cap T^*$. Il existe alors des mots x et y dans L_k et une règle r dans R_R tels que w soit obtenu par la recombinaison suivante : $(x, y) \vdash_r (w, z)$. Nous remarquons que la deuxième propriété de D implique que la règle r appartient à l'ensemble R_R, car, dans le cas contraire, w n'aurait pas fait partie de l'alphabet terminal T. La même propriété de D implique que $x, y \in L_k \setminus T^*$. Par conséquent, $x, y \in M_k$. De même, il existe, par construction, une règle $(r, out, out) \in R'$ et nous pouvons recombiner x et y dans Π à l'aide de r en envoyant les mots résultants w et z à l'extérieur de la membrane.

Inversement, si un mot terminal $w \in T^*$ est envoyé à l'extérieur de la membrane par la règle (r, out, out), où $r \in R_R$, ce w peut être obtenu en utilisant la règle r dans D. D'autre part, si un mot w est envoyé à l'extérieur de la membrane par une règle (r, out, out), où $r \notin R_R$, ce mot contiendra des lettres non-terminales grâce à la deuxième propriété de D.

□

Maintenant pour prouver le théorème 9.1.6 il suffit d'observer que le système distribué à changement de phase construit dans le théorème 6.3.1 satisfait les conditions du lemme précédent.

Il est possible aussi d'utiliser la méthode des molécules assistantes pour démontrer le théorème 9.1.6. Nous pouvons reprendre le schéma du calcul présenté à la figure 4.1, voir page 46, et introduire les molécules assistantes nécessaires. Par exemple, en suivant les notations du théorème 6.3.1, voir figure 6.2 à la page 79, nous considérons la molécule assistante Z_2Y_i qui doit apparaître avec une période

de 8. Pour cela il suffit d'avoir les mots Z_2Y_i, Z_2Z, Z_2^kZ, $1 \leq k \leq 7$ parmi les axiomes, ainsi que les règles $\frac{Z_2^{k-1} \mid Y_i}{Z_2^k \mid Z} \begin{matrix}(here)\\(here)\end{matrix}$, $\frac{Z_2^7 \mid Y_i}{Z_2 \mid Z} \begin{matrix}(here)\\(here)\end{matrix}$, $\frac{Z_2^kY_i \mid \varepsilon}{Z_2^kY_i \mid \varepsilon} \begin{matrix}(out)\\(out)\end{matrix}$ et $\frac{Z_2Y_i \mid \varepsilon}{Z_2Y_i \mid \varepsilon} \begin{matrix}(out)\\(out)\end{matrix}$, $1 \leq k \leq 7$ parmi les règles de la membrane.

9.1.3 Systèmes de type 2b, 2c et 2a

Dans cette section nous montrons que les systèmes à membranes de type 2b et 2c produisent tous les langages récursivement énumérables.

Théorème 9.1.8. *Soit $D = (V, T, A, R)$ un système distribué à changement de phase de degré 1. Il existe un système à membranes avec recombinaison de type 2b et de degré 1, Π, qui simule D et pour lequel $L(\Pi) = L(D)$.*

Démonstration. Considérons $\Pi = (V, T, [_1]_1, A, R')$, où

$$R' = \{(r, here, here); (r, out, out); Out(r) : r \in R\} \cup \left\{ \frac{w \mid \varepsilon}{w \mid \varepsilon} \begin{matrix}(out)\\(out)\end{matrix} : w \in A \right\}.$$

Il est facile de voir que Π simule D. Soient $x, y \in L_k$ qui sont complémentaires par rapport à la règle $r \in R$. On a l'application suivante dans D : $(x, y) \vdash_r (w, z)$. Dans Π, nous avons $(x, y) \vdash_r (w, z)$ et les mots w et z sont envoyés à l'extérieur de la membrane, tout en gardant leurs copies à l'intérieur. Les mots x et y sont envoyés à l'extérieur par $Out(r)$ et il ne restent plus dans la membrane. On obtient donc $L(D) \subseteq L(\Pi)$.

Il est clair que l'inclusion inverse est vraie. Si nous utilisons une suite des règles $(r_1, here, here), \ldots, (r_n, out, out)$ pour obtenir une molécule w dans Π, la suite homologue $r_1, \ldots, r_n$ permet d'obtenir le même mot w dans D.

□

Comme $VDH_1 = RE$ nous obtenons :

Corollaire 9.1.9. *Les systèmes à membranes avec recombinaison de type 2b et de degré 1 produisent tous les langages récursivement énumérables.*

Pour les systèmes de type 2c nous avons un résultat similaire.

Théorème 9.1.10. *Les systèmes à membranes avec recombinaison de type 2c et de degré 1 produisent tous les langages récursivement énumérables.*

Plus précisément, ce théorème est un corollaire du théorème 9.1.6. Si nous reconsidérons la démonstration de ce théorème, nous observons que la situation quand une copie de w reste dans la membrane ne peut pas se produire.

Par contre, nous n'avons pas trouvé la puissance d'expression des systèmes de type 2a avec une seule membrane, mais nous supposons qu'ils n'ont pas une grande puissance d'expression.

Conjecture 9.1.1. $ELSP_1(spl, in_w) = REG$.

On voit que ce cas est très similaire à celui de la section 9.1.1. En effet, il est facile de voir que

$M_{k+1} = M_k \cup \{\sigma_{h_h}(M_k) \setminus \sigma_{h_o}(M_k)\}$,

en utilisant les notation de cette section-là.

De même, il est facile de voir que si une molécule w est produite dans la membrane avant d'être envoyée à l'extérieur, elle y restera pour toujours. Dans le cas contraire elle ne sera jamais présente dans la membrane. Ce point est l'argument principal de notre conjecture.

9.2 Systèmes à membranes avec recombinaison non-étendus

Dans cette section nous considérons les systèmes à membranes avec recombinaison non-étendus qui différent des systèmes introduits auparavant par le fait qu'ils ne possèdent pas d'alphabet terminal. Tous les mots envoyés à l'extérieur du système forment donc le résultat du calcul.

Nous notons par $LSP_m(spl, in)$ la famille des langages produits par les systèmes à membranes non-étendus de degré au plus m. Nous n'indiquons pas le type du système considéré, car, comme le montre le théorème suivant, une membrane ne suffit plus pour obtenir une grande puissance d'expression et avec deux membranes nous pouvons éviter les problèmes qui nous ont obligé d'introduire les différentes définitions.

Théorème 9.2.1. $LSP_1(spl, in) \subset REG$.

Ce théorème est un corollaire du lemme 9.1.2 et du théorème 3.1.3, voir aussi le tableau 2.1.

Théorème 9.2.2. *Soit $G = (N, T, S, P)$ une grammaire arbitraire. Il existe un système à membranes avec recombinaison non-étendu de degré 2, Π, ayant des règles globales qui simule G et pour lequel $L(\Pi) = L(G)$.*

Démonstration. Posons $\Pi = (V, [_1[_2\,]_2]_1, M_1, M_2, R_1 \cup R_2)$.

Soient $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(a_n = B)$ et $B \notin \{N \cup T\}$. Supposons aussi que $T = \{a_s, \ldots, a_{n-1}\}$.

Dans ce qui suit nous supposons que :
$1 \leq \mathbf{i} \leq n$, $0 \leq \mathbf{j} \leq n-1$, $\mathbf{a} \in N \cup T \cup \{B\}$, $s \leq \mathbf{p} \leq n-1$, $s-1 \leq \mathbf{q} \leq n-2$.

L'alphabet V est défini par :
$V = N \cup T \cup \{B\} \cup \{X, Y, X_\mathbf{i}, Y_\mathbf{i}, X'_\mathbf{j}, Y'_\mathbf{j}, X_0, Y_0, X', Y', Z_1, Z_2, V, W, V_\mathbf{i}, W_\mathbf{i}, V'_\mathbf{j}, W'_\mathbf{j}, V_{s-1}, W_{s-1}, V', W', V_E, W_E, V'_E, W'_E, V''_E, E, Z_V, Z_W\}$.

Les axiomes sont définis par :
$M_1 = \{XSBY\} \cup \{Z_1 v Y_\mathbf{j} : \exists u \to v a_\mathbf{j} \in P\} \cup \{Z_1 Y_\mathbf{i}, Z_1 Y_0, Z_1 Y'_\mathbf{j}, X' Z_1, X Z_1, X_V B Z_1, Z_1 Y'_W, X'_V Z_1, Z_1 W_p, Z_1 W'_q, Z_1 W_{s-1}, V' Z_1, V Z_1, V_E Z_1, Z_1 W'_E, V'_E Z_1, V''_E Z_1, Z_W, Z_V\}$.

$M_2 = \{X_\mathbf{i} a_\mathbf{i} Z_2, X'_\mathbf{j} Z_2, X_\mathbf{j} Z_2, X_0 Z_2, Z_2 Y', Z_2 Y, Z_2 Y_W, Z_2 W, V Z_2, V_\mathbf{p} a_\mathbf{p} Z_2, V'_\mathbf{q} Z_2, V_\mathbf{q} Z_2, V_{s-1} Z_2, Z_2 W', Z_2 W, Z_2 W_E, Z_2 W''_E, E Z_2\}$.

Les règles du système sont définis de la manière suivante.
Règles de R_1 :

$$1.1 : \frac{\varepsilon \mid uY \ (in)}{Z \mid vY \ (in)}, \quad \text{si } \exists u \to v \in P;$$

$$1.2 : \frac{\varepsilon \mid a_i Y \ (in)}{Z \mid Y_i \ (here)}; \quad 1.3 : \frac{\mathbf{a} \mid Y_i \ (in)}{Z_1 \mid Y'_{i-1} \ (here)}; \quad 1.4 : \frac{\mathbf{a} \mid Y'_j \ (in)}{Z_1 \mid Y_j \ (here)};$$

$$1.5 : \frac{X_0 \mid \mathbf{a} \ (here)}{X' \mid Z_1 \ (in)}; \quad 1.6 : \frac{X' \mid \mathbf{a} \ (here)}{X \mid Z_1 \ (in)};$$

$$1.7 : \frac{X' \mid \mathbf{a} \ (here)}{X_V B \mid Z_1 \ (in)}; \quad 1.8 : \frac{\mathbf{a} \mid Y_W \ (here)}{Z_1 \mid Y'_W \ (in)}; \quad 1.9 : \frac{X_V \mid B \ (here)}{X'_V \mid Z_1 \ (in)};$$

$$1.10 : \frac{X'_V \mid B \ (in)}{X''_V \mid Z_1 \ (here)}; \quad 1.11 : \frac{\mathbf{a} \mid Y''_W \ (in)}{Z_1 \mid W \ (here)};$$

$$1.12 : \frac{\varepsilon \mid a_p W \ (in)}{Z_1 \mid W_p \ (here)}; \quad 1.13 : \frac{\mathbf{a} \mid W_p \ (in)}{Z_1 \mid W'_{p-1} \ (here)}; \quad 1.14 : \frac{\mathbf{a} \mid W'_q \ (in)}{Z_1 \mid W_q \ (here)};$$

$$1.15 : \frac{V_{s-1} \mid \mathbf{a} \ (here)}{V' \mid Z_1 \ (in)}; \quad 1.16 : \frac{V' \mid \mathbf{a} \ (here)}{V \mid Z_1 \ (in)};$$

$$1.17 : \frac{V' \mid \mathbf{a} \ (here)}{V_E \mid Z_1 \ (in)}; \quad 1.18 : \frac{\mathbf{a} \mid W_E \ (here)}{Z_1 \mid W'_E \ (in)}; \quad 1.19 : \frac{V_E \mid \mathbf{a} \ (here)}{V'_E \mid Z_1 \ (in)};$$

$$1.20 : \frac{V'_E \mid \mathbf{a} \ (in)}{V''_E \mid Z_1 \ (here)}; \quad 1.21 : \frac{\mathbf{a} \mid W''_E \ (in)}{Z_W \mid \varepsilon \ (in)};$$

$$1.22 : \frac{E \mid \mathbf{a} \ (in)}{\varepsilon \mid Z_V \ (out)};$$

Règles de R_2 :

$$2.1 : \frac{X \mid \mathbf{a} \ (here)}{X_i a_i \mid Z_2 \ (out)}; \quad 2.2 : \frac{X_i \mid \mathbf{ab} \ (here)}{X'_{i-1} \mid Z_2 \ (out)}; \quad 2.3 : \frac{X'_j \mid \mathbf{a} \ (here)}{X_j \mid Z_2 \ (out)};$$

$$2.4 : \frac{\mathbf{a} \mid Y_0 \ (out)}{Z_2 \mid Y' \ (here)}; \quad 2.5 : \frac{\mathbf{a} \mid Y' \ (out)}{Z_2 \mid Y \ (here)};$$

$$2.6 : \frac{\mathbf{a} \mid BY' \ (out)}{Z_2 \mid Y_W \ (here)}; \quad 2.7 : \frac{\mathbf{a} \mid Y'_W \ (out)}{Z_2 \mid Y''_W \ (here)}; \quad 2.8 : \frac{X''_V \mid B \ (here)}{V \mid Z_2 \ (out)};$$

$$2.9 : \frac{V \mid \mathbf{a} \ (here)}{V_p a_p \mid Z_2 \ (out)}; \quad 2.10 : \frac{V_p \mid \mathbf{ab} \ (here)}{V'_{p-1} \mid Z_2 \ (out)}; \quad 2.11 : \frac{V'_q \mid \mathbf{a} \ (here)}{V_q \mid Z_2 \ (out)};$$

$$2.12 : \frac{\mathbf{a} \mid W_{s-1} \ (out)}{Z_2 \mid W' \ (here)}; \quad 2.13 : \frac{\mathbf{a} \mid W' \ (out)}{Z_2 \mid W \ (here)};$$

$$2.14 : \frac{\mathbf{a} \mid BW' \ (out)}{Z_2 \mid W_E \ (here)}; \quad 2.15 : \frac{\mathbf{a} \mid W'_E \ (out)}{Z_2 \mid W''_E \ (here)}; \quad 2.16 : \frac{V''_E \mid \mathbf{a} \ (here)}{E \mid Z_2 \ (out)};$$

Nous affirmons que $L(\Pi) = L(G)$.

Nous allons prouver cette affirmation de la manière suivante. Nous montrerons comment il est possible de simuler les dérivations de la grammaire formelle G et,

par conséquent, nous obtiendrons que $L(G) \subseteq L(\Pi)$. Dans le même temps, nous montrerons que toutes les autres possibilités de calcul n'aboutissent pas à un mot terminal, ce qui prouvera l'énoncé.

Notre simulation est fondée sur la méthode «faire-tourner-et-simuler», voir aussi la section 4.1.2 et le théorème 4.3.1. Nous effectuons la rotation et la simulation des règles de la grammaire par les règles 1.1 à 1.6 et 2.1 à 2.5. Lorsque nous avons B à la fin du mot, nous substituons les parenthèses X et Y par V et W à l'aide des règles 1.7 à 1.11 et 2.6 à 2.8. Puis nous effectuons une étape de rotation, mais nous faisons tourner le mot d'une lettre uniquement si cette lettre est une lettre terminale de la grammaire. Les règles 1.12 à 1.16 et 2.9 à 2.13 permettent de réaliser ce procédé. Quand on a à nouveau le symbole B en fin de mot, nous savons que le mot considéré est terminal et qu'il est de la forme correcte. À cet instant, nous éliminons les parenthèses V et W par les règles 2.14 à 2.16 et 1.17 à 1.22. Nous observons que, par construction, ces règles permettent d'envoyer à l'extérieur du système des mots terminaux uniquement.

Nous observons aussi que les règles sont symétriques par rapport aux parenthèses X, Y et V, W : les règles 1.2 à 1.11 et 2.1 à 2.8 sont identiques aux règles 1.12 à 1.21 et 2.9 à 2.16 si on effectue la transformation suivante : $X \to V$, $Y \to W$, $X_V \to V_E$, $Y_W \to W_E$ et $0 \to s - 1$. Cette symétrie est due au fait que nous effectuons deux fois des étapes similaires : la rotation et la fixation du résultat. La différence est seulement dans le fait que la deuxième fois nous considérons uniquement des lettres terminales pour la rotation.

Nous remarquons que ce procédé de répétition de la rotation a été proposé par Gh. Păun dans [43], mais le système obtenu possédait quatre membranes.

Notations

Nous observons que le site inférieur de chaque règle représente une molécule qui est déjà présente dans la membrane correspondante. De même, un des résultats de la recombinaison contiendra Z et ne pourra plus participer à la production du résultat. Nous omettrons donc ces molécules et nous écrirons :

$Xwa_iY \underset{1.2}{\longrightarrow} XwY_i(tar_1)$ à la place de

$(Xw|a_iY, Z|Y_i) \vdash_{1.2} (XwY_i, Za_iY)\ (tar_1; tar_2)$.

Comme dans la section précédente les règles sont construites de telle sorte que si une molécule ne s'accorde avec aucune règle, elle ne pourra jamais participer à une recombinaison et ne contribuera donc pas au résultat. Nous écrivons $w\uparrow$ si la molécule w est dans une telle situation.

Nous observons aussi que, par construction, les règles de R_1 peuvent être utilisées uniquement dans la première membrane, car elles contiennent au moins un indicateur de cible *in*, et les règles de R_2 sont utilisables dans la deuxième membrane uniquement, car elles contiennent au moins un indicateur de cible *out*.

Rotation

Nous commençons avec Xwa_kY dans la première membrane.

$Xwa_kY \underset{1.2}{\longrightarrow} XwY_k(in)$,
$XwY_k \underset{2.1}{\longrightarrow} \mathbf{X_j a_j w Y_k}(out)$.

Nous avons maintenant 4 cas :

a) Nous avons des molécules de la forme $X_0a_kwY_i, i > 0$, dans la première membrane. Dans ce cas :
$X_0a_kwY_i \underset{1.5}{\longrightarrow} X'a_kwY_i(in) \uparrow$.

b) Nous avons des molécules de la forme $X_ia_kwY_0, i > 0$, dans la première membrane. À nouveau elles ne peuvent plus évoluer :
$X_ia_kwY_0 \uparrow$.

c) Nous avons des molécules de la forme $X_ja_kwY_i$, $i, j > 0$, dans la première membrane. Nous avons le calcul suivant :
$X_ja_kwY_i \underset{1.3}{\longrightarrow} X_ja_kwY'_{i-1}(in)$,
$X_ja_kwY'_{i-1} \underset{2.2}{\longrightarrow} X'_{j-1}a_kwY'_{i-1}(out)$,
$X'_{j-1}a_kwY'_{i-1} \underset{1.4}{\longrightarrow} X'_{j-1}a_kwY_{i-1}(in)$,
$X'_{j-1}a_kwY_{i-1} \underset{2.3}{\longrightarrow} \mathbf{X_{j-1}a_kwY_{i-1}}(out)$ ou $X'_{j-1}a_kwY_0 \underset{2.4}{\longrightarrow} X'_{j-1}a_kwY'(out) \uparrow$.
Nous avons décrémenté simultanément les indices de X et Y.

d) Nous avons des molécules de la forme $X_0a_kwY_0$, dans la première membrane. Ceci nous donne :
$X_0a_kwY_0 \underset{1.5}{\longrightarrow} X'a_kwY_0(in)$,
$X'a_kwY_0 \underset{2.4}{\longrightarrow} X'a_kwY'(out)$,
(*) $X'a_kwY' \underset{1.6}{\longrightarrow} Xa_kwY'(in)$,
(**) $Xa_kwY' \underset{2.5}{\longrightarrow} \mathbf{Xa_kwY}(out)$ ou $Xa_kwY' \underset{2.1}{\longrightarrow} X_ia_ia_kwY'(out) \uparrow$.

Nous avons donc fait tourner le mot Xwa_kY d'une lettre. Il est possible d'appliquer les règles 1.7 et 2.6 dans les cas marqués par (*) et (**), mais ces applications seront discutées plus tard.

Simulation des productions de la grammaire

Si nous avons un mot $XwuY$ dans la première membrane et s'il existe une production $u \to v \in P$, nous pouvons appliquer la règle 1.1 qui produit $XwvY$, c'est-à-dire qu'on a simulé la production correspondante de la grammaire.

Contrôle de la terminaison

Nous obtenons $X'wY'$ dans la première membrane (cas (*)). À cet instant nous pouvons utiliser les règles suivantes :
$X'wY' \underset{1.7}{\longrightarrow} X_VBwY'(in)$,
$X_VBwY' \underset{2.6}{\longrightarrow} X_VBw'Y_W(out)$ si $w = w'B$,
$X_VBw'Y_W \underset{1.8}{\longrightarrow} X_VBw'Y'_W(here)$ ou $X_VBw'Y_W \underset{1.9}{\longrightarrow} X'_VBw'Y_W(in) \uparrow$,
$X_VBw'Y'_W \underset{1.9}{\longrightarrow} X'_VBw'Y'_W(in)$,
$X'_VBw'Y'_W \underset{2.7}{\longrightarrow} X'_VBw'Y''_W(out)$,

$X'_V Bw'Y''_W \underset{1.10}{\longrightarrow} X''_V Bw'Y''_W(here)$ ou $X'_V Bw'Y''_W \underset{1.11}{\longrightarrow} X'_V Bw'W(in)\uparrow$,
$X''_V Bw'Y''_W \underset{1.11}{\longrightarrow} X''_V Bw'W(in)$,
$X''_V Bw'W \underset{2.8}{\longrightarrow} \mathbf{VBw'W}(out)$.
L'évolution de $VBw'W$ sera examiné plus bas.
Autres évolutions possibles :
1. (cas (**)) $XwY' \underset{2.6}{\longrightarrow} Xw'Y_W(out)$ $(w = w'B)$,
$Xw'Y_W \underset{1.8}{\longrightarrow} Xw'Y'_W(here)\uparrow$.
2. $X_V BwY' \underset{2.5}{\longrightarrow} X_V BwY(out)$,
$X_V Bw'a_iY \underset{1.2}{\longrightarrow} X_V Bw'Y_i(in)\uparrow$ ou $X_V BwY \underset{1.9}{\longrightarrow} X'_V wY(in)\uparrow$.

Rotation des terminaux

Maintenant nous effectuons la rotation des lettres terminales de la grammaire uniquement, c'est-à-dire nous contrôlons si $w' \in T^*$. Le calcul est très similaire à la rotation générale, voir plus haut, et si on substitue X par V, Y par W et 0 par $s-1$, on obtient la même évolution.

Nous avons Vwa_kW dans la première membrane.
$Vwa_kW \underset{1.12}{\longrightarrow} VwW_k(in)$,
$VwW_k \underset{2.9}{\longrightarrow} \mathbf{V_j a_j w W_k}(out)$.
Nous avons maintenant 4 cas :
a) Nous avons des molécules de la forme $V_{s-1}a_kwW_i, i > s-1$, dans la première membrane. Dans ce cas :
$V_{s-1}a_kwW_i \underset{1.15}{\longrightarrow} V'a_kwW_i(in)\uparrow$.
b) Nous avons des molécules de la forme $V_ia_kwW_{s-1}, i > s-1$, dans la première membrane. À nouveau elles ne peuvent plus évoluer :
$V_ia_kwW_{s-1}\uparrow$.
c) Nous avons des molécules de la forme $V_ja_kwW_i$, $i,j > s-1$, dans la première membrane. Nous avons le calcul suivant :
$V_ja_kwW_i \underset{1.13}{\longrightarrow} V_ja_kwW'_{i-1}(in)$,
$V_ja_kwW'_{i-1} \underset{2.10}{\longrightarrow} V'_{j-1}a_kwW'_{i-1}(out)$,
$V'_{j-1}a_kwW'_{i-1} \underset{1.14}{\longrightarrow} V'_{j-1}a_kwW_{i-1}(in)$,
$V'_{j-1}a_kwW_{i-1} \underset{2.11}{\longrightarrow} \mathbf{V_{j-1}a_k w W_{i-1}}(out)$ ou $V'_{j-1}a_kwW_0 \underset{2.12}{\longrightarrow} V'_{j-1}a_kwW'(out)\uparrow$.
Nous avons décrémenté simultanément les indices de V et W.
d) Nous avons des molécules de la forme $V_{s-1}a_kwW_{s-1}$, dans la première membrane. Ceci nous donne :
$V_{s-1}a_kwW_{s-1} \underset{1.15}{\longrightarrow} V'a_kwW_{s-1}(in)$,
$V'a_kwW_{s-1} \underset{2.12}{\longrightarrow} V'a_kwW'(out)$,
$(+)$ $V'a_kwW' \underset{1.16}{\longrightarrow} Va_kwW'(in)$,
$(++)$ $Va_kwW' \underset{2.13}{\longrightarrow} \mathbf{Va_k w W}(out)$ ou $Va_kwW' \underset{2.10}{\longrightarrow} V_ia_ia_kwW'(out)\uparrow$.

Nous avons donc fait tourner le mot Vwa_kW d'une lettre. Il est possible d'appliquer les règles 1.17 et 2.15 dans les cas marqués par (+) et (++), mais ces applications seront discutées plus tard.

Obtention du résultat

Nous obtenons $V'wW'$ dans la première membrane (cas (+)). À cet instant nous pouvons utiliser les règles suivantes :

$V'wW' \underset{1.17}{\longrightarrow} V_EwW'(in)$,

$V_EwY' \underset{2.14}{\longrightarrow} X_Vw'W_E(out)$ si $w = w'B$,

$V_Ew'W_E \underset{1.18}{\longrightarrow} V_Ew'W'_E(here)$ ou $V_Ew'W_E \underset{1.19}{\longrightarrow} V'_Ew'W_E(in)\uparrow$,

$V_Ew'W'_E \underset{1.19}{\longrightarrow} V'_Ew'W'_E(in)$,

$V'_Ew'W'_E \underset{2.15}{\longrightarrow} V'_Ew'W''_E(out)$,

$V'_Ew'W''_E \underset{1.20}{\longrightarrow} V''_Ew'W''_E(here)$ ou $V'_Ew'W''_E \underset{1.21}{\longrightarrow} V'_Ew'(in)\uparrow$,

$V''_Ew'W''_E \underset{1.21}{\longrightarrow} V''_Ew'(in)$,

$V''_Ew' \underset{2.16}{\longrightarrow} Ew'(out)$,

$Ew' \underset{1.22}{\longrightarrow} \mathbf{w}'(out)$.

Dans ce cas, le mot w' qui est terminal est envoyé à l'extérieur et il fera partie du résultat du calcul.

Autres évolutions possibles :

1. (cas (++)) $VwW' \underset{2.14}{\longrightarrow} Vw'W_E(out)$ $(w = w'B)$,

$Vw'W_E \underset{1.18}{\longrightarrow} Vw'W'_E(here)\uparrow$.

2. $V_EwW' \underset{2.13}{\longrightarrow} V_EwW(out)$,

$V_Ew'a_pW \underset{1.12}{\longrightarrow} V_Ew'W_p(in)\uparrow$ ou $V_EwW \underset{1.19}{\longrightarrow} V'_EwW(in)\uparrow$.

Il est facile de voir qu'en suivant le calcul ci-dessus nous obtenons tous les mots de $L(G)$ et, comme nous avons considéré tous les cas possibles, il est clair que le système ne produit pas d'autres mots.

□

9.3 Systèmes à membranes avec recombinaison et communication immédiate

Dans cette section nous considérons les systèmes à membranes avec recombinaison et communication immédiate qui différent des systèmes introduits dans la section 9.1 par le fait que le résultat de chaque recombinaison doit partir de la membrane dans laquelle il a été produit. Plus précisément, on peut voir un système à membranes avec recombinaison et communication immédiate comme un système à membranes avec recombinaison dont les règles ne possèdent pas d'indicateur de cible *here* et, dans le même temps, chaque règle de recombinaison apparaît avec les quatre combinaisons possibles des indicateurs de cible *in* et *out*.

Nous notons par $ELSP_m(spl, move)$ la famille des langages produits par les systèmes à membranes avec recombinaison et communication immédiate de degré au plus m. Comme dans la section précédente nous n'indiquons pas le type du système considéré, car nous pouvons éviter les problèmes qui nous ont obligé à examiner différentes variantes.

Si on a une seule membrane, à chaque étape du calcul nous envoyons les mêmes molécules à l'extérieur lesquelles sont effectivement le résultat de la recombinaison des axiomes. Par conséquent, on a le résultat suivant :

Théorème 9.3.1. $ELSP_1(spl, move) = FIN$.

Deux membranes suffisent déjà pour avoir une grande puissance de calcul.

Théorème 9.3.2. *Soit $G = (N, T, S, P)$ une grammaire arbitraire. Il existe un système à membranes avec recombinaison et communication immédiate de degré 2, Π, qui simule G et pour lequel $L(\Pi) = L(G)$.*

Démonstration. Posons $\Pi = (V, T, [_1[_2\,]_2]_1, M_1, M_2, R_1, R_2))$.

Soient $N \cup T \cup \{B\} = \{a_1, a_2, \ldots, a_n\}$ $(a_n = B)$ et $B \notin \{N \cup T\}$.

Dans ce qui suit nous supposons que :

$1 \leq \mathbf{i} \leq n$, $0 \leq \mathbf{j} \leq n-1$, $1 \leq \mathbf{k} \leq n$, $\mathbf{a} \in N \cup T \cup \{B\}$.

L'alphabet V est défini par :

$V = N \cup T \cup \{B\} \cup \{X, Y, X_\mathbf{i}, Y_\mathbf{i}, X_0, Y_0, X'_\mathbf{j}, Y'_\mathbf{j}, X', Y', Z, Z_X, Z_Y\}$.

L'alphabet terminal T est celui de la grammaire formelle G.

Les axiomes sont définis par :

$M_1 = \{XSBY\} \cup \{ZvY_\mathbf{j} \mid u \to va_\mathbf{j} \in P\} \cup \{ZY_\mathbf{i}, ZY_0, ZY'_\mathbf{j}, X'Z, XZ, Z_X\}$.

$M_2 = \{X_\mathbf{i}a_\mathbf{i}Z, X'_\mathbf{j}Z, X_\mathbf{j}Z, ZY', ZY, Z_Y\}$.

Les règles du système sont définis de la manière suivante.

Règles de R_1 :

$$1.1 : \frac{\varepsilon \mid uY}{Z \mid vY_\mathbf{j}}, \quad \text{si } \exists u \to va_\mathbf{j} \in P; \qquad 1.1' : \frac{\varepsilon \mid a_\mathbf{i}uY}{Z \mid Y_\mathbf{i}}, \quad \text{si } \exists u \to \varepsilon \in P;$$

$$1.2 : \frac{\varepsilon \mid a_\mathbf{i}Y}{Z \mid Y_\mathbf{i}}; \qquad 1.3 : \frac{\mathbf{a} \mid Y_\mathbf{k}}{Z \mid Y'_{\mathbf{k}-1}}; \qquad 1.4 : \frac{\mathbf{a} \mid Y'_\mathbf{j}}{Z \mid Y_\mathbf{j}};$$

$$1.5 : \frac{X_0 \mid \mathbf{a}}{X' \mid Z}; \qquad 1.6 : \frac{X' \mid \mathbf{a}}{X \mid Z}; \qquad 1.7 : \frac{X' \mid \mathbf{a}}{\varepsilon \mid Z_X}.$$

Règles de R_2 :

$$2.1 : \frac{X \mid \mathbf{a}}{X_\mathbf{i}a_\mathbf{i} \mid Z}; \qquad 2.2 : \frac{X_\mathbf{k} \mid \mathbf{a}}{X'_{\mathbf{k}-1} \mid Z}; \qquad 2.3 : \frac{X'_\mathbf{j} \mid \mathbf{a}}{X_\mathbf{j} \mid Z}; \qquad 2.4 : \frac{\mathbf{a} \mid Y_0}{Z \mid Y'};$$

$$2.5 : \frac{\mathbf{a} \mid Y'}{Z \mid Y}; \qquad 2.6 : \frac{\mathbf{a} \mid BY_0}{Z_Y \mid \varepsilon}.$$

Nous affirmons que $L(\Pi) = L(G)$.

Nous allons prouver cette affirmation de la manière suivante. Nous montrerons comment il est possible de simuler les dérivations de la grammaire formelle G et, par conséquent, nous obtiendrons que $L(G) \subseteq L(\Pi)$. Dans le même temps, nous montrerons que toutes les autres possibilités de calcul n'aboutissent pas à un mot terminal, ce qui prouvera l'énoncé.

Notre simulation est fondée sur la méthode «faire-tourner-et-simuler», voir aussi la section 4.1.2 et le théorème 4.3.1. Nous commençons avec le mot Xwa_iY dans la première membrane. Après l'application de la règle 1.2, le mot XwY_i est envoyé dans la deuxième membrane. Puis la règle 2.1 est appliquée et la première membrane reçoit les mots $X_ja_jwY_i$ $(1 \leq j \leq n)$. À partir de cet instant, le système Π fait décroître les indices i et j simultanément. Les mots pour lesquels $j \neq i$ sont éliminés et seuls les mots $X_0a_iwY_0$ restent. Puis, nous obtenons le mot Xa_iwY, nous avons donc fait tourner le mot Xwa_iY d'une lettre. Si $a_iw = a_iw'B$ et $a_iw' \in T^*$, ce dernier mot faisant partie du résultat est aussi produit et envoyé à l'extérieur du système.

Nous allons souligner certaines des similitudes entre le système construit et les systèmes distribués à changement de phase, en particulier le système D_G présenté au théorème 4.3.1.

- Les membranes peuvent être considérés comme composants.
- Seul le résultat de la recombinaison participe au calcul dans le composant suivant, respectivement dans la membrane suivante.
- Si un mot ne possède pas de molécule complémentaire par rapport à une règle, il est éliminé. Dans le cas du système Π ce mot ne pourra jamais participer à une recombinaison et, par conséquent, ne pourra pas contribuer au résultat.

Pour obtenir le troisième point nous avons construit les règles d'une manière particulière : le site inférieur de chaque règle représente une molécule qui est déjà présente dans la membrane correspondante. Par conséquent, si une molécule ne s'accorde avec aucune règle, elle ne pourra jamais participer à une recombinaison et ne contribuera donc pas au résultat. Nous écrivons $w\uparrow$ si la molécule w est dans une telle situation.

Il y a cependant des différences entre le système que nous venons de construire et le système D_G :

- Si une molécule possède une molécule complémentaire par rapport à une règle, cette molécule restera pour toujours dans la membrane correspondante et participera à chaque étape à une recombinaison.

Comme nous simulons une grammaire, à chaque étape nous avons plusieurs molécules qui codent des branches différentes de la simulation et qui évoluent en parallèle. C'est pourquoi notre système est construit d'une manière qui permet de gérer ce parallélisme. Par conséquent, les différences ci-dessus ne sont pas importantes pour notre démonstration.

La démonstration est donc très similaire à la démonstration du théorème 4.3.1 et le calcul du système Π suit de près le calcul fait par le système D_G.

En vue des mêmes arguments que ceux utilisés pendant la démonstration du théorème 9.2.2, nous écrivons :

$Xwa_iY \underset{1.2}{\longrightarrow} XwY_i$ à la place de

$(Xw|a_iY, Z|Y_i) \vdash_{1.2} (XwY_i, Za_iY)$.

Nous remarquons que la molécule résultante est envoyée immédiatement hors de la membrane dans laquelle elle a été produite. À chaque étape elle change donc de membrane. Par conséquent, il n'est pas nécessaire d'indiquer la membrane courante, car l'indication du numéro de la règle suffit pour lever toutes les ambiguïtés.

Rotation

Nous commençons avec Xwa_kY dans la première membrane.

$Xwa_kY \underset{1.2}{\longrightarrow} XwY_k$,

$XwY_k \underset{2.1}{\longrightarrow} \mathbf{X_ja_jwY_k}$.

Nous avons maintenant 4 cas :

a) Nous avons des molécules de la forme $X_0a_kwY_i, i > 0$, dans la première membrane. Dans ce cas :

$X_0a_kwY_i \underset{1.5}{\longrightarrow} X'a_kwY_i \uparrow$.

b) Nous avons des molécules de la forme $X_ia_kwY_0, i > 0$, dans la première membrane. Comme dans le cas précédant elles ne peuvent plus évoluer :

$X_ia_kwY_0 \uparrow$.

c) Nous avons des molécules de la forme $X_ja_kwY_i$, $i, j > 0$, dans la première membrane. Nous avons le calcul suivant :

$X_ja_kwY_i \underset{1.3}{\longrightarrow} X_ja_kwY'_{i-1}$,

$X_ja_kwY'_{i-1} \underset{2.2}{\longrightarrow} X'_{j-1}a_kwY'_{i-1}$,

$X'_{j-1}a_kwY'_{i-1} \underset{1.4}{\longrightarrow} X'_{j-1}a_kwY_{i-1}$,

$X'_{j-1}a_kwY_{i-1} \underset{2.3}{\longrightarrow} \mathbf{X_{j-1}a_kwY_{i-1}}$, ou

$X'_{j-1}a_kwY_0 \underset{2.4}{\longrightarrow} X'_{j-1}a_kwY' \uparrow$ $(X'_{j-1}a_kwY_0 \underset{2.6}{\longrightarrow} X'_{j-1}a_kw \uparrow)$.

Nous avons décrémenté simultanément les indices de X et Y.

d) Nous avons des molécules de la forme $X_0a_kwY_0$, dans la première membrane. Ceci nous donne :

$X_0a_kwY_0 \underset{1.5}{\longrightarrow} X'a_kwY_0$,

(*) $X'a_kwY_0 \underset{2.4}{\longrightarrow} X'a_kwY'$,

(**) $X'a_kwY' \underset{1.6}{\longrightarrow} Xa_kwY'$,

$Xa_kwY' \underset{2.5}{\longrightarrow} \mathbf{Xa_kwY}$ ou $Xa_kwY' \underset{2.1}{\longrightarrow} X_ia_ia_kwY' \uparrow$.

Nous avons donc fait tourner le mot Xwa_kY d'une lettre. Il est possible d'appliquer les règles 2.6 et 1.7 dans les cas marqués par (*) et (**), mais ces applications seront discutées plus tard.

Simulation des productions de la grammaire

Si nous avons le mot $XwuY$ dans la première membrane et si la règle $u \to v$ existe dans P, nous pouvons appliquer la règle 1.1 ou 1.1′ pour simuler la production correspondante de la grammaire et puis nous continuons comme décrit plus haut.

Obtention du résultat

Nous obtenons $X'a_kwY_0$ dans la deuxième membrane (cas (*)). À cet instant nous pouvons utiliser les règles suivantes :

$X'a_kw'BY_0 \underset{2.6}{\longrightarrow} X'a_kw'$ $(w = w'B)$,

$X'a_kw' \underset{1.7}{\longrightarrow} a_kw'$.

Dans ce cas si $a_kw' \in T^*$, il fera partie du résultat.

Nous pouvons aussi appliquer la règle 1.6 :

$X'a_kw' \underset{1.6}{\longrightarrow} Xa_kw'$,

$Xa_kw' \underset{2.1}{\longrightarrow} X_ia_ia_kw' \uparrow$.

Autres évolutions possibles :

Nous avons le mot $X'a_kwY'$ dans la première membrane (cas (**)).

$X'a_kwY' \underset{1.7}{\longrightarrow} a_kwY'$,

$a_kwY' \underset{2.5}{\longrightarrow} a_kwY$,

$a_kw'a_iY \underset{1.2}{\longrightarrow} a_kw'Y_i \uparrow$.

Il est facile de voir qu'en suivant le calcul ci-dessus nous obtenons tous les mots de $L(G)$ et, comme nous avons considéré tous les cas possibles, il est clair que le système ne produit pas d'autres mots. De plus, le calcul suit de près les calculs des systèmes présentés aux théorèmes 4.3.1 et 9.2.2.

□

9.4 Conclusions

Dans cette section nous réunissons certains traits communs des systèmes construits précédemment dans ce chapitre.

Nous observons que tous les systèmes de ce chapitre éliminent les molécules non-désirées par deux procédés. Le premier procédé consiste à envoyer ces molécules à l'extérieur du système, tandis que le deuxième les garde dans les membranes d'une manière qui ne permet pas leur utilisation par la suite. Pour atteindre ce but, les règles sont faites d'une manière particulière et si une molécule ne s'accorde avec aucune règle, elle ne pourra jamais participer à une recombinaison et, par conséquent, elle ne pourra jamais contribuer au résultat. Ces éliminations que nous appelons externes et respectivement, internes, jouent un rôle important pendant le calcul.

Nous remarquons aussi que les systèmes présentés dans ce chapitre possèdent des liens forts avec les systèmes distribués à changement de phase. En effet, le calcul dans les systèmes à membranes avec recombinaison de ce chapitre, modulo l'élimination interne, suit de près le calcul dans les systèmes distribués à changement de phase correspondants.

Finalement, la méthode des molécules assistantes est applicable pour les systèmes qui permettent une élimination externe puissante, plus précisément les systèmes à membranes avec recombinaison de type 1, 2b et 2c.

Conclusions

Dans ce travail nous avons étudié des systèmes de Head et leur extensions ou, plus généralement, des divers systèmes fondés sur l'opération de la recombinaison. Nous avons vu que cette opération est très puissante et seulement des petits ajouts sont nécessaires pour obtenir la puissance d'expression d'une machine de Turing. Ceci n'est pas étonnant, car l'opération de la recombinaison présente des éléments dépendants du contexte et la dépendance du contexte dans la théorie des langages formels est presque l'universalité.

Nous avons vu que, sans ajouter d'autres ingrédients, l'opération de la recombinaison permet de produire des langages rationnels uniquement. Nous avons étudié pour la première fois la relation entre la recombinaison simple et la recombinaison double et nous avons montré que la famille des langages de recombinaison double est strictement incluse dans la famille des langages de recombinaison simple. Nous avons trouvé aussi des exemples non-triviaux de langages rationnels qui ne peuvent pas être des langages de recombinaison.

Puis nous avons étudié les systèmes distribués à changement de phase qui sont à la fois simples et puissants. Nous avons montré que ces systèmes ont une grande puissance d'expression avec deux composants et même avec un seul composant. Nous avons aussi montré à l'aide d'un logiciel que nous avons créé auparavant que certains articles sur les systèmes distribués à changement de phase possèdent des erreurs et nous avons montré comment il est possible de les corriger.

Le point central de cette thèse, la méthode des molécules assistantes, nous a permis d'uniformiser les démonstrations de plusieurs résultats en les rendant beaucoup plus simples. Les systèmes distribués à changement de phase, les systèmes distribués à changement de phase étendus, les systèmes de tubes à essai avec filtres alternants, les systèmes de tubes à essai modifiés et même les systèmes à membranes avec recombinaison ont constitué le champ d'application de cette méthode que nous avons introduite. En généralisant, nous pouvons dire que l'opération de la recombinaison en combinaison avec l'élimination de molécules dans le sens décrit à la fin du chapitre 6, suffisent pour atteindre la puissance d'expression d'une machine de Turing. De cette manière, nous récapitulons certaines réflexions sur la recombinaison faites par différents auteurs.

De même, l'opération de la recombinaison en combinaison avec les membranes donne un outil très puissant et il n'est pas étonnant qu'on arrive facilement à obtenir une grande puissance d'expression.

Ce travail laisse plusieurs problèmes ouverts. D'abord, nous avons fait un petit apport à la résolution du problème de la caractérisation des langages de recombinaison. Comme le montre notre travail, il faut considérer désormais deux cas : les langages de recombinaison simple et les langages de recombinaison double.

Nous avons aussi montré que les systèmes distribués à changement de phase sont de bons candidats pour une « machine de Turing » dans le domaine des calculs moléculaires. Comme nous l'avons vu, plusieurs systèmes présentent des similitudes avec les systèmes distribués à changement de phase ou peuvent même se réduire à ceux-ci. Nous croyons que des études dans cette direction pourront apporter beaucoup de résultats intéressants. C'est pourquoi nous avons aussi présenté au chapitre 4 un système distribué à changement de phase universel qui a une description très compacte. Nous continuerons des recherches dans cette direction afin de trouver, d'une manière semblable à ce qui a été fait pour les machines de Turing, des systèmes qui seront de plus en plus petits. Nous croyons que la recherche des systèmes minimaux en combinaison avec la structure et le fonctionnement simple des systèmes distribués à changement de phase peuvent apporter des résultats intéressants.

Une autre direction de recherche consiste dans l'étude et le développement de la méthode des molécules assistantes. Comme c'est une méthode très générique, nous croyons qu'elle pourra avoir beaucoup d'applications pour différents systèmes fondés sur la recombinaison, mais aussi pour d'autres types d'opérations n-aires, avec $n > 2$. Par exemple, comme nous avons clos le chapitre des systèmes à membranes avec recombinaison en présentant des résultats définitifs, nous espérons appliquer la méthode des molécules assistantes à d'autres types de systèmes à membranes qui ne sont pas fondés sur la recombinaison.

Nous nous sommes aussi rapproché de la résolution du problème de la puissance d'expression des systèmes de tubes à essai avec deux tubes. Les variantes que nous avons proposé différent de moins en moins de la définition originelle, c'est pourquoi nous espérons que le travail que nous avons fait apportera de la lumière sur ce problème ouvert. Ces variantes présentent aussi des propriétés intéressantes comme les poubelles fixes et mobiles dont l'étude n'est pas encore terminée.

Pour conclure, nous voulons ajouter que l'étude des systèmes d'inspiration biologique est très fascinante et elle peut apporter des gains colossaux. C'est peut-être le moment d'apprendre des nouvelles choses et en reformulant Wordsworth, [57], « que la nature soit notre professeur ».

Bibliographie

[1] L. M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, **266**, (1994), 1021–1024.

[2] P. Bonizzoni, C. D. Felice, G. Mauri, and R. Zizza. Regular languages generated by reflexive finite splicing systems. In Z. Esik and Z. Fulop, editors, *Proceedings of Developments in Language Theory 2003, DLT03, Szeged, Hungary, July 7-11, 2003.* 2003, volume 2710 of *Lecture Notes in Computer Science*, 134–145.

[3] P. Bonizzoni, C. D. Felice, G. Mauri, and R. Zizza. The structure of reflexive finite splicing languages via Schützenberger constants, 2004. Submitted.

[4] J. Cocke and M. Minsky. Universality of tag systems with p=2. *Journal of the ACM*, **11**(1), (1964), 15–20.

[5] E. Csuhaj-Varjú, L. Kari, and G. Păun. Test tube distributed systems based on splicing. *Computers and AI*, **15**(2–3), (1996), 211–232.

[6] K. Culik II and T. Harju. Splicing semigroups of dominoes and DNA. *Discrete Applied Mathematics*, **31**, (1991), 261–277.

[7] F. Freund, R. Freund, M. Margenstern, M. Oswald, Y. Rogozhin, and S. Verlan. P systems with cutting/recombination rules assigned to membranes. In A. Alhazov, C. Martín-Vide, and G. Păun, editors, *Preproceedings of the Workshop on Membrane Computing. Tarragona, July 17-22, 2003.* 2003, 241–251.

[8] R. Freund and F. Freund. Test tube systems : when two tubes are enough. In G. Rozenberg and W. Thomas, editors, *Preproceedings of DLT99, Developments in Language Theory.* World Scientific Publishing Company, 2000, 275–286.

[9] P. Frisco. On two variants of splicing super-cell systems. *Romanian Journal of Information Science and Technology*, 4(1-2), (2001), 89–100.

[10] P. Frisco and C. Zandron. On variants of communicating distributed H systems. *Fundamenta Informaticae*, **48**(1), (2001), 9–20.

[11] T. E. Goode Laun. *Constants and Splicing Systems.* Ph.D. thesis, Binghamton University, 1999.

[12] T. Head. Formal language theory and DNA : an analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, **49**(6), (1987), 737–759.

[13] T. Head. Splicing languages generated with one sided context. In G. Păun, editor, *Computing with Bio-Molecules. Theory and Experiments.* Springer Verlag, Berlin, Heidelberg, New York, 1998, 158–181.

[14] T. Head, G. Păun, and D. Pixton. Language theory and molecular genetics. generative mechanisms suggested by DNA recombination. In *Handbook of Formal Languages, 3 volumes* [47], 295–360.

[15] J. Hopcroft, R. Motwani, and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Mass., 2nd edition, 2001.

[16] L. Kari. DNA computing : arrival of biological mathematics. *The Mathematical Intelligencer*, **19**(2), (1997), 9–22. Earlier version under the title DNA computers, tomorrow's reality. Bulletin of the European Association for Theoretical Computer Science, (59) :256–266, June 1996 `http ://www.csd.uwo.ca/ lila/amsn.ps`.

[17] S. M. Kim. An algorithm for identifying spliced languages. In *Proceedings of Cocoon97.* 1997, volume 1276 of *Lecture Notes in Computer Science*, 403–411.

[18] S. Kleene. *Introduction to Metamathematics.* Van Nostrand Comp. Inc., New-York, 1952.

[19] R. J. Lipton. Using DNA to solve NP-complete problems. *Science*, **268**, (1995), 542–545.

[20] V. Manca. A proof of regularity for finite splicing. In N. Jonoska, G. Paun, and G. Rozenberg, editors, *Aspects of Molecular Computing. Essays Dedicated to Tom Head on the Occasion of His 70th Birthday*, Springer Verlag, Berlin, Heidelberg, New York, volume 2950 of *LNCS.* 2004, 309–317.

[21] M. Margenstern and Y. Rogozhin. Generating all recursively enumerable languages with a time-varying distributed H system of degree 2. Technical report, Institut Universitaire de Technologie de Metz, 1999. Publications du G.I.F.M.

[22] M. Margenstern and Y. Rogozhin. A universal time-varying distributed H-system of degree 2. In L. Kari, H. Rubin, and D. H. Wood, editors, *Proceedings of the 4th DIMACS meeting on DNA based computers.* Elsevier, 1999, volume 52, (1–3), 73–80.

[23] M. Margenstern and Y. Rogozhin. About time-varying distributed H systems. In A. Condon and G. Rozenberg, editors, *DNA Computing : 6th International Workshop on DNA-Based Computers, DNA 2000, Leiden, The Netherlands, June 13-17, 2000, Revised Papers.* Springer Verlag, Berlin, Heidelberg, New York, 2000, volume 2054 of *Lecture Notes in Computer Science*, 53–62.

[24] M. Margenstern and Y. Rogozhin. Time-varying distributed H systems of degree 2 generate all recursively enumerable languages. In Martin-Vide and Mitrana [31], 399–407.

[25] M. Margenstern and Y. Rogozhin. Extended time-varying distributed H systems - universality result. In *Proceedings of The 5th World Multi-Conference on Systemics, Cybernetics and Informatics, Industrial Systems, SCI 2001, Orlando, Florida USA, July 22-25, 2001.* 2001, volume IX.

[26] M. Margenstern and Y. Rogozhin. Time-varying distributed H systems of degree 1 generate all recursively enumerable languages. In M. Ito, G. Păun, and S. Yu,

editors, *Words, Semigroups, and Transductions*, World Scientific, Singapore. 2001, 329–340. Festschrift in Honor of Gabriel Thierrin.

[27] M. Margenstern and Y. Rogozhin. A universal time-varying distributed H system of degree 1. In N. Jonoska and N. C. Seeman, editors, *DNA Computing : 7th International Workshop on DNA-Based Computers, DNA7, Tampa, FL, USA, June 10-13, 2001. Revised Papers.* Springer Verlag, Berlin, Heidelberg, New York, 2002, volume 2340, 371–380.

[28] M. Margenstern, Y. Rogozhin, and S. Verlan. Time-varying distributed H systems of degree 2 can carry out parallel computations. In M. Hagiya and A. Ohuchi, editors, *DNA Computing : 8th International Workshop on DNA-Based Computers, DNA8, Sapporo, Japan, June 10-13, 2002. Revised Papers.* Hokkaido University, Springer Verlag, Berlin, Heidelberg, New York, 2002, volume 2568 of *Lecture Notes in Computer Science*, 326–336.

[29] M. Margenstern, Y. Rogozhin, and S. Verlan. Time-varying distributed H systems with parallel computations : the problem is solved. In *Preliminary Proceedings of DNA Computing, 9th international Workshop on DNA-Based Computers, DNA 2003, Madison, Wisconsin, USA, 1-4 June 2003.* 2003, 47–51.

[30] C. Martin-Víde, G. Păun, and A. Rodríguez-Patón. P systems with immediate communication. *Romanian Journal of Information Science and Technology*, 4(1-2), (2001), 171–182.

[31] C. Martin-Vide and V. Mitrana, editors. *Where Do Mathematics, Computer Science and Biology Meet.* Kluwer Academic, Dortrecht, 2000.

[32] M. Minsky. *Computations : Finite and Infinite Machines.* Prentice Hall, Englewood Cliffts, NJ, 1967.

[33] D. Pixton. Regularity of splicing languages. *Discrete Applied Mathematics*, **69**(1–2), (1996), 101–124.

[34] L. Priese, Y. Rogozhin, and M. Margenstern. Finite H-systems with 3 test tubes are not predictable. In R. Altman, A. Dunker, L. Hunter, and T. Klein, editors, *Proceedings of Pacific Symposium on Biocomputing, 3, Kapalua, Maui, January 1998, Hawaii, USA.* World Scientific Publishing Company, 1998, 547–558.

[35] A. Păun. On time-varying H systems. *Bulletin of EATCS*, **67**, (1999), 157–164.

[36] A. Păun and M. Păun. On the membrane computing based on splicing. In Martin-Vide and Mitrana [31], 409–422.

[37] G. Păun. On the splicing operation. *Discrete Applied Mathematics*, **70**(1), (1996), 57–79.

[38] G. Păun. Regular extended H systems are computationally universal. *JALC*, **1**(1), (1996), 27–36.

[39] G. Păun. DNA computing : distributed splicing systems. In J. Mycielsky, G. Rozenberg, and A. Salomaa, editors, *Structures in Logic and Computer Science. A Selection of Essays in Honor of A. Ehrenfeucht.* Springer Verlag, Berlin, Heidelberg, New York, 1997, volume 1261 of *Lecture Notes in Computer Science*, 351–370.

[40] G. Păun. DNA computing based on splicing : universality results. In M. Margenstern, editor, *Proceedings of the Second Internetional Colloquium on Universal Machines and Computations, Metz, France.* IUT de Metz, 1998, volume I, 67–91.

[41] G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, **1**(61), (2000), 108–143. Also TUCS Report No. 208, 1998.

[42] G. Păun. DNA computing based on splicing : universality results. *Theoretical Computer Science*, **231**(2), (2000), 275–296. Journal version of [40].

[43] G. Păun. *Membrane Computing. An Introduction.* Springer Verlag, Berlin, Heidelberg, New York, 2002.

[44] G. Păun, G. Rozenberg, and A. Salomaa. *DNA Computing : New Computing Paradigms.* Springer Verlag, Berlin, Heidelberg, New York, 1998.

[45] G. Păun and T. Yokomori. Membrane computing based on splicing. In E. Winfree and D. K. Gifford, editors, *DNA Based Computers V.* American Mathematical Society, 1999, volume 54 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 217–232.

[46] Y. Rogozhin. Small universal turing machines. *Theoretical Computer Science*, **168**(2), (1996), 215–240.

[47] G. Rozenberg and A. Salomaa. *Handbook of Formal Languages, 3 volumes.* Springer Verlag, Berlin, Heidelberg, New York, 1997.

[48] M. P. Schutzenberger. Sur certaines opérations de fermeture dans les langages rationnels. *Symposia Mathematica*, **15**, (1975), 245–253.

[49] TVDHsim: Time-varying distributed H systems simulator. `http://lita.sciences.univ-metz.fr/~verlan/`.

[50] S. Verlan. On extended time-varying distributed H systems. In *Preproceedings at 6th International Workshop on DNA-Based Computers, DNA 2000, Leiden, The Netherlands, June 13–17, 2000.* 2000, 281. Poster.

[51] S. Verlan. *Calculs Moléculaires : les Systèmes Distribués à Changement de Phase.* Master's thesis, Université de Metz, 2001.

[52] S. Verlan. On enhanced time-varying distributed H systems. *Computer Science Journal of Moldova*, **10**(3), (2002), 263–279. Kishinev.

[53] S. Verlan. About splicing P systems with immediate communication and non-extended splicing P systems. In A. Alhazov, C. Martín-Vide, and G. Păun, editors, *Preproceedings of the Workshop on Membrane Computing. Tarragona, July 17-22, 2003.* 2003, 461–473.

[54] S. Verlan. A frontier result on enhanced time-varying distributed H systems with parallel computations. In *Preproceedings of DCFS'03, Descriptional Complexity of Formal Systems, Budapest, Hungary, July 12-14, 2003.* 2003, 221–232.

[55] S. Verlan. Communicating distributed H systems with alternating filters. In N. Jonoska, G. Paun, and G. Rozenberg, editors, *Aspects of Molecular Computing. Essays Dedicated to Tom Head on the Occasion of His 70th Birthday,*

Springer Verlag, Berlin, Heidelberg, New York, volume 2950 of *LNCS*. 2004, 367–384.

[56] S. Verlan and R. Zizza. 1-splicing vs. 2-splicing : separating results. In *Proceedings of WORDS'03, Turku, Finland, September 10-13, 2003*. 2003, 320–331.

[57] W. Wordsworth. The tables turned. In *Wordworth's Poems*, Ed. P. Wayne, Dent, London, volume 1996. 1965.

[58] C. Zandron. The P systems web page. `http://psystems.disco.unimib.it/`.

[59] C. Zandron, C. Ferretti, and G. Mauri. A reduced distributed splicing system for RE languages. In G. Păun and A. Salomaa, editors, *New trends in Formal Language. Control, cooperatio, and conbinatorics*. Springer Verlag, Berlin, Heidelberg, New York, 1997, volume 1218 of *Lecture Notes in Computer Science*, 346–366.

[60] C. Zandron, C. Ferretti, and G. Mauri. Nine test tubes generate any RE language. *Theoretical Computer Science*, **231**(2), (2000), 171–180.

[61] R. Zizza. *On the power of classes of splicing systems*. Ph.D. thesis, University of Milan, 2002.

Résumé

Ce travail se situe au coeur d'un domaine récent : les calculs (bio)moléculaires qui se place à la rencontre de la biologie et de l'informatique. La recherche dans ce domaine porte sur des modèles de calcul ayant une inspiration biologique.

Les systèmes de Head, fondés sur l'opération de la recombinaison, ainsi que leurs extensions représentent des modèles de calcul parmi les plus importants du domaine.

Nous nous sommes concentrés sur l'étude de différentes extensions des systèmes de Head en présentant souvent des résultats d'universalité optimaux par rapport aux paramètres principaux de ces systèmes ce qui a permis d'établir plusieurs frontières entre la décidabilité et l'indécidabilité pour ces extensions. De plus, nous avons trouvé certains traits communs de ces systèmes et nous avons conçu une méthode générique permettant de démontrer facilement des équivalences avec des modèles de calcul classiques.

Nous avons aussi considéré les systèmes à membranes qui sont inspirés par la structure et le fonctionnement de la cellule vivante. Nous nous sommes particulièrement intéressés à la combinaison de ces modèles et les systèmes de Head : les systèmes à membranes avec recombinaison et nous avons résolu une série des problèmes ouverts pour ces systèmes.

Ce travail manipule des notions fondamentales de la théorie de la calculabilité et du calcul moléculaire et donne des solutions à certains problèmes importants de ce dernier domaine en posant dans le même temps plusieurs questions intéressantes.

Abstract

This work is situated in the heart of a recent domain : (bio)molecular computing, which is a domain where biology and computer science meet. This domain investigates models of computing having biological inspiration.

Head splicing systems, based on the splicing operation, as well as their extensions are one of the most important models of computing in this domain.

We concentrated on the study of different extensions of Head systems by showing often optimal universality results with respect to main parameters of these systems what permitted us to establish several frontiers between decidability and undecidability for these extensions. Moreover, we found several common features of these systems and we designed a generic method that permits to prove easily equivalences with classical models of computing.

We also considered membrane systems that are inspired by the structure and the functioning of a living cell. We were particularly interested in the combination of these models and Head systems : splicing membrane systems and we solved a number of open problems for these systems.

This work manipulates fundamental notions of the theory of computing and of the molecular computing. It gives solutions to certain important open problems of the last domain and it poses at the same time several interesting questions.