N° d'ordre : 3105

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **SEVERINE FRATANI**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

AUTOMATES À PILES DE PILES ... DE PILES

Soutenue le : 14 décembre 2005

Après avis des rapporteurs :

Rapporteurs :

Jean Berstel	Professeur émérite
Serge Grigorieff	Professeur

Devant la commission d'examen composée de :

Jean Berstel	Professeur émérite	Rapporteur
Frédérique Carrère .	Maître de Conférence	Directrice de thèse
Didier Caucal	Chargé de recherches CNRS, HDR	Examinateur
Serge Grigorieff	Professeur	Rapporteur
Géraud Sénizergues	Professeur	Directeur de thèse
Alexandre Zvonkine	Professeur	Examinateur

– 2005 –

Remerciements

Avant de parler des gens qui ont directement influencé mon travail de thèse, j'aimerais avoir un mot pour souligner l'environnement accueillant que j'ai trouvé en venant faire ma maîtrise à l'université Bordeaux I. Après une errance dans diverses universités, c'est ici que j'ai ressenti le plus de passion, de motivation et d'envie de transmettre. Le choix d'effectuer mon DEA, puis ma thèse à Bordeaux à été fortement influencé par les rencontres que j'ai faites au LaBRI et par l'ambiance chaleureuse y régnant.

Mes premières pensées vont à mes directeurs de thèse Frédérique Carrère et Géraud Sénizergues qui m'accompagnent et me guident depuis mon DEA, ainsi que pour leur famille auxquelles j'ai volé un nombre incalculable de débuts de soirées. Géraud et Frédérique ont toujours oeuvré à m'encourager, me soutenir et me motiver dès que nécessaire et l'ambiance amicale et détendue de nos réunions de travail ont contribué à faire de moi une thésarde heureuse. J'ai été tout autant comblée sur le plan de la compétence scientifique et de l'acharnement du travail bien fait. L'accompagnement dont j'ai eu la chance de bénéficier portera ces fruits certainement bien au-delà de l'obtention du doctorat. J'espère avoir acquis à leurs cotés un petit peu de la rigueur scientifique et de la passion qu'ils ont tenté de me transmettre. En particulier, merci à Frédérique pour les lectures et relectures de mes interminables écrits, et à Géraud pour son accueil à n'importe quel moment de la journée, pour avoir invariablement une réponse à chacune de mes questions, et un nombre incommensurable de nouvelles idées et pistes de travail.

Je remercie mes rapporteurs Jean Berstel et Serge Grigorieff pour l'intérêt qu'il ont accepté de porter à ma thèse. C'est une grande fierté que de recevoir leur avis sur mon travail. Je remercie également les membres du jury Didier Caucal et Alexandre Zvonkine de me faire l'amitié et l'honneur de leur présence.

J'aimerais également avoir un mot pour tous ceux qui ont contribué à l'élaboration de ma thèse : Mireille Bousquet-Mélou et Bruno Courcelle pour avoir pris le temps de répondre à nos questions, Julien Bernet, Olivier Ly et Igor Walukiewicz pour m'avoir écoutée et conseillée, David Janin pour l'intérêt qu'il porte à mon travail depuis mon DEA, ainsi pour son soutien financier par le biais du RTN Games, à Pascal Weil qui m'a poussée à entamer la rédaction de ce manuscrit, et a toujours été disponible, bienveillant et de sage conseil. J'applaudis mes acolytes de bureau Roland Barillot, Alexis Bouquet et Xavier Briand pour m'avoir supportée et avoir si souvent réglé mes mésententes avec les machines. Merci aussi à tous ceux que j'oublie, ce n'est qu'étourderie de ma part.

Enfin, un grand merci à ma famille, ils savent que je n'en serais pas là sans eux, et combien je leur suis reconnaissante. Je n'oublie pas tous ceux qui m'ont entouré pendant ces trois années, en particulier Christophe qui m'a écoutée patiemment des soirées entières, et qui est depuis mon arrivée à Bordeaux d'un soutien et d'une amitié sans failles.

Introduction

Théorie des langages et des automates

Les origines de la théorie des langage se trouvent bien antérieurement aux débuts de l'informatique, dans la tentative de modélisation des langues naturelles. Elle prend tout son essor avec l'apparition des langages de programmation puisqu'elle s'avère tout à fait adaptée à leur description et leur analyse. Outre cette application majeur, les nombreuses connections avec divers domaines, rendent cette théorie incontournable en informatique. De plus, la richesse des objets permettant d'aborder les langages (systèmes de réécriture, grammaires, automates, ou d'autres outils moins mécaniques tels que les théories algébriques ou la logique) font qu'elle continue à être étudiée pour elle-même.

Dans les années 60, Büchi utilise les automates de mots infinis pour la résolution de problèmes mathématiques en mettant en relation les langages réguliers et des formules logiques exprimant des propriétés portant sur les entiers naturels [Büc60]. Rabin généralise les résultats de Büchi à des structures munies de plusieurs relations successeurs [Rab69]. Les liens entre langages et logique sont depuis communément utilisés et occupent encore actuellement une place centrale, dans des domaines très théoriques, mais également dans des théories plus appliquées telles que le contrôle et la vérification. Ainsi, même si dans la pratique, les langages algébriques sont suffisants à la formalisation des langages de programmation, l'étude de langages plus complexes ouvre des voies dans divers domaines : logique, algorithmique,...

Langages de niveau supérieur

Le célèbre linguiste Chomsky définit une hiérarchie de classes de langages sur trois niveaux : le langages réguliers (reconnus par automates finis) constituent le niveau 0, le niveau 1 est formé des langages algébriques (reconnus par automates à piles) et le niveau 2 correspond aux langages récursivement énumérables (reconnus par machines de Turing). Bien que les langages algébriques s'avèrent suffisant à reconnaître des langages d'expressions arithmétiques ou de formules booléennes, leurs limites apparaissent dès qu'il s'agit de modéliser des langages faisant apparaître des phénomènes de "copie". Ainsi, un exemple typique de langage qui ne soit pas algébrique est l'ensemble des mots "doublés" : $w \cdot w$. Il est toutefois raisonnable d'imaginer qu'il n'est pas nécessaire d'utiliser la puissance d'une machine de Turing pour reconnaître ce type de langage relativement simple.

Aho introduit dans [Aho68] les "grammaires indexées", extensions des grammaires algébriques, permettant d'insérer strictement une nouvelle classe entre les algébriques et le récursivement énumérables. Ces grammaires permettent, par exemple, de générer le langage $\{a^n b^n c^n \mid n \geq 0\}$ qui constitue un exemple classique de langage qui ne soit pas algébrique, ou encore le langage des mots "doublés". Il définit ensuite une classe équivalente d'automates : les "nested stack automata"- [Aho69]. Les langages définis par Aho bénéficient des mêmes propriétés générales que les langages algébriques. Plus tard, Maslov [Mas74, Mas76], donne un nouveau type de grammaires beaucoup plus digeste, caractérisant les même langages, et se prêtant naturellement à la généralisation à un niveau quelconque. Il introduit simultanément une classe d'automate équivalente : les "multilevel stack automata", définissant ainsi une hiérarchie infinie stricte de langages dont le niveau 0 est formé des langages réguliers, le niveau 1 correspond aux langages algébriques et le niveau infini est celui des langages récursivement énumérables.

Soulignons que l'idée de tels automates avait été une première fois évoquée par Greibach en 70 dans le cadre de "familles abstraites d'automates" [Gre70]. Parallèlement, des grammaires appelées "macro-grammaires", équivalentes aux '"grammaires indexées' de Aho sont introduites dans [Fis68], la même année que ces dernières. Ces grammaires ont ensuite étés étendues en une hiérarchie : la "OI-hiérarchie", correspondant à la hiérarchie des langages de niveaux supérieurs définie par Maslov, et déjà évoquée dans [Mas74]. Ces extensions des macro-grammaires, ainsi que leurs liens avec les automates à piles d'ordres supérieurs furent principalement étudiées par Damm [Dam82, DG86], puis Engelfriet [Eng83, ES84, EV86, Eng91].

Après une période relativement creuse, l'intérêt pour les langages et automates à piles de niveaux supérieurs s'est fortement réveillé. En 2000, Knapik, Niwinsky et Urzyczyn étudient la logique monadique sur les arbres infinis générés par les "automates d'ordre supérieur" [KNU02], variante épurée des automates étudiés par les précédents auteurs. Parallèlement, Caucal définit par itération de deux opérations simples sur les graphes, une hiérarchie de graphes admettant une théorie monadique décidable. De nombreux travaux ont lié cette hiérarchie, aux graphes de calculs des automates d'ordre supérieur [Cac03, CW03]. Dans [FS03], les automates sont utilisés à la résolutions de problèmes logiques et combinatoires liés aux suites d'entiers naturels et rationnels (voir aussi également la Partie VI de ce manuscrit).

Piles itérées et automates

Une pile sur un alphabet A_1 est une liste de symboles dans A_1. Trois opérations permettent d'utiliser une pile : la lecture du symbole de tête de pile, la suppression du symbole de tête de pile et l'empilement d'un symbole donné en tête de pile.

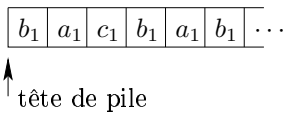


FIG. 1 – Une pile

Un manière de considérer des piles sur un ensemble de symboles infini est de travailler avec des *piles de piles*. Une 2-pile sur des alphabets (A_1, A_2) est une liste de couples (symbole dans A_2, pile sur A_1). Cette construction peut être itérée autant de fois que l'on veut. En l'appliquant k fois, nous obtenons l'ensemble des k-piles. Une k-pile est constituée d'une liste de couples (symbole de A_k, $(k-1)$-pile sur $(A_1, \ldots, A_{k-1})$). La figure 2 donne la représentation d'une 3-pile sur des alphabets A_1, A_2, A_3. Les symboles de l'alphabet i sont indicés par i.

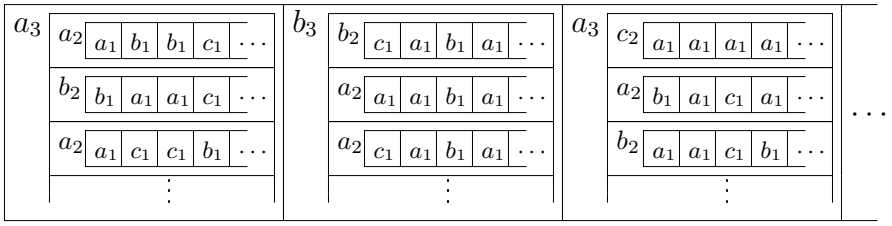


FIG. 2 – Une 3-pile :ω

Les opérations généralement appliquées à une k-pile sont les suivantes :
- la lecture des k symboles en tête de chaque niveau du sommet de la pile : effectuée par l'opération **top**
- l'empilement d'un symbole a_i au niveau i, avec recopie de la pile de niveau $i-1$ la plus au sommet : effectué par l'opération $\mathbf{push}_{a_i}$
- le dépilement le tête de niveau i : correspondant à l'application de $\mathbf{pop}_i$
- le remplacement du symbole de tête de niveau i par une lettre a_i : c'est l'opération $\mathbf{change}_{a_i}$.

Illustrons l'effet de ces différentes opérations sur notre exemple ω. La liste de symboles de tête de ω est

$$\mathrm{top}(\omega) = a_3 a_2 a_1$$

L'empilement d'un symbole $a_1 \in A_1$ a pour effet d'ajouter ce symbole au sommet de la pile de niveau 1 la plus en tête de ω. Le résultat de l'instruction $\mathrm{push}_{a_1}(\omega)$ est schématisé par la figure 3.

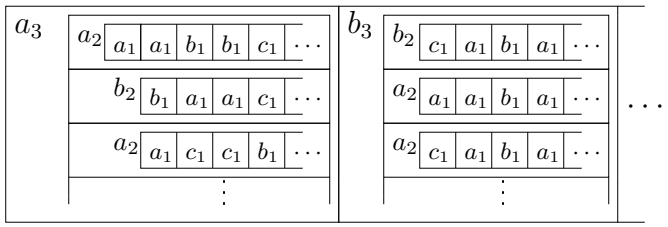


FIG. 3 – Résultat de $\mathrm{push}_{a_1}(\omega)$

L'empilement d'un symbole $a_i \in A_i$ pour $i > 1$ s'accompagne d'une copie de la pile de niveau $i-1$ la plus en tête, comme représenté figure 4 pour l'empilement du symbole b_2 : la 1-pile de tête est recopiée en tête et le nouveau symbole de tête associé est b_2.

L'application de pop_i à une k-pile (pour $i \in [1, k]$) a pour effet de dépiler le premier élément de la pile du niveau i se trouvant en tête de pile, c'est à dire, de supprimer la pile de niveau

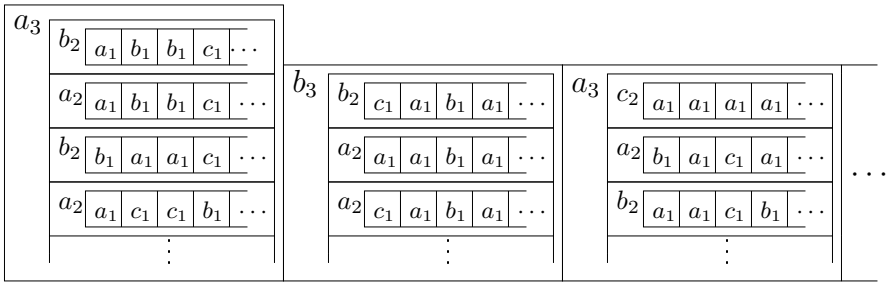


FIG. 4 – Résultat de $\text{push}_{b_2}(\omega)$

$i-1$ située en sommet de pile. Ainsi pop_1 efface le premier symbole de la pile de niveau 1 la plus en tête. Le résultat de $\text{pop}_1(\omega)$ est représenté sur la figure 5. La figure 6 illustre le résultat de $\text{pop}_3(\omega)$: la pile de niveau 2 constituant le sommet de ω est entièrement effacée.

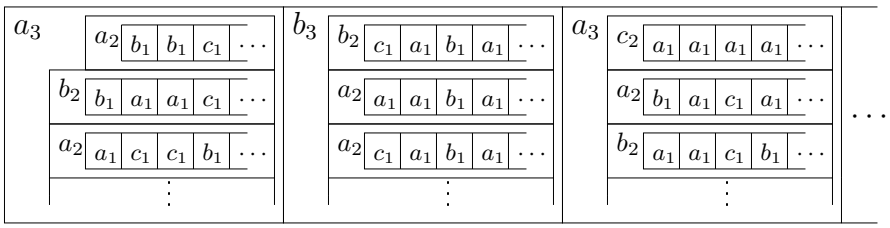


FIG. 5 – Résultat de $\text{pop}_1(\omega)$

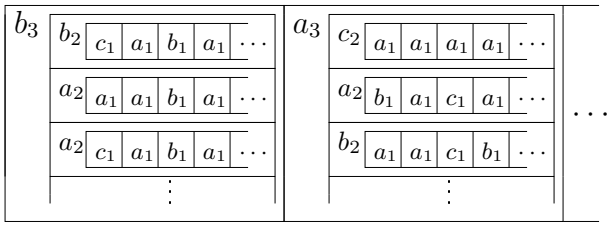


FIG. 6 – Résultat de $\text{pop}_3(\omega)$

Enfin, l'application de change_{a_i} pour $a_i \in A_i$ a pour effet de remplacer le symbole de tête de niveau i par le nouveau symbole a_i. Le résultat de l'application de change_{b_2} à notre exemple ω est représenté sur la figure 7.

Un *automate à k-piles* est une machine utilisant une k-pile comme mémoire. La seule information dont peut disposer la machine sur l'état de sa mémoire est la liste des symboles de sommet de pile, qu'elle obtient en interrogeant le résultat de l'application de top sur la pile courante. Les modifications de la mémoire sont effectuées par le biais des instructions de type push, pop et change. Le choix de l'instruction à appliquer est déterminé en fonction de l'*état* de la machine et la liste des symboles de tête de pile.

Ce sont ces automates, ainsi que diverses extensions et restrictions que nous nous proposons d'étudier dans ce manuscrit.

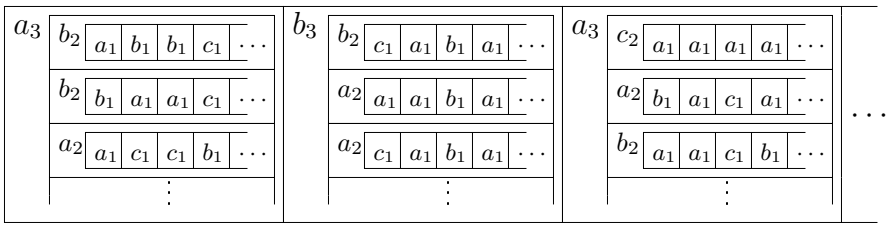


FIG. 7 – Résultat de $\text{change}_2(\omega)$

Résumé

Les principales contributions apportées dans ce manuscrit sont de trois natures et font respectivement l'objet des parties II, III et IV.

Logique

Nous proposons une étude des propriétés de la Logique du Second Ordre Monadique (LSOM) sur la structure associée au type "piles k-itérées". Son domaine est l'ensemble des k-piles, et ses relations correspondent aux graphes des instructions de types push, pop et change. Nous prouvons que cette structure admet une théorie SOM décidable, que les ensembles définissables (en LSOM) sont exactement les ensembles générés par un automate à k-piles contrôle par des propriétés définissables, et que toute SOM-formule satisfiable admet une solution *définissable*. Cette dernière propriété que nous nommons "propriété du Modèle Définissable" (MD), est essentielle à l'obtention de ces résultats et n'avait, à notre connaissance, pas encore été explicitement étudiée.

Ces résultats sont obtenus par le biais d'une étude générale des propriétés de la logique SOM sur des structures d'arbres étiquetés

$$\langle \{0,1\}^*, \varepsilon, \text{SUCC}_0, \text{SUCC}_1, O_1, \ldots, O_m \rangle$$

où SUCC_i est la relation binaire "successeur par la lettre i" et les O_j sont des relations unaires formant un étiquetage des sommets de l'arbre. Nous donnons une méthode permettant le transfert de propriétés d'un graphe étiqueté particulier vers un tel arbre étiqueté : transfert de la décidabilité de la théorie SOM et de la propriété MD. Nous définissons également une classe d'automates *avec oracles* reconnaissant exactement les langages définissables en LSOM dans l'arbre étiqueté.

Ces méthodes permettent donc d'obtenir de nombreux étiquetages $\vec{O}$, pour lesquels les propriétés de l'arbre étiqueté par $\vec{O}$ généralisent les résultats obtenus par Rabin sur l'arbre simple.

Théorie des automates

Différentes classes d'automates reconnaissant les langages de niveaux supérieurs ont été décrites au cours du temps, et ceci, sans véritables comparaisons en termes de déterminisme et de graphe de calcul. Nous établissons les liens entre ces divers classes d'automates, ainsi qu'avec certaines classes d'automates *contrôlés* (i.e., dont l'application des transitions dépend d'un test d'appartenance sur la pile courante). Toutes ces comparaisons s'appliquent également aux

classes d'automates déterministes. Nous prouvons principalement que les langages de niveau k (i.e., reconnus par automate à k-piles non contrôlé) sont exactement les langages reconnus par :

- automates à k-piles contrôlés par des propriétés définissable en LSOM (dans la structure décrite précédemment)
- automates à k-piles "à instructions symétriques", c'est à dire de type $\text{push}_{i,a}$ et $\overline{\text{push}}_{i,a}$, où $\overline{\text{push}}_{i,a}$ correspond à l'instruction inverse de $\text{push}_{i,a}$ (c'est à dire qu'elle n'est applicable qu'au piles dont le sommet contient deux copies identiques de la même $(i-1)$-pile
- automates à $(k-\ell)$-piles dont le niveau 1 est contrôlé par un automate à ℓ-piles déterministe.

Un cas particulier du dernier point est donc que tout automate à $(k+1)$-piles (déterministe) correspond à un automate à 1-piles (déterministe) dont la pile est contrôlée par un automate à k-piles déterministe.

Algorithmique

Nous proposons de résoudre des problèmes algorithmiques liés aux suites de nombres entiers et rationnels par le biais des automates à piles itérées. Nous décrivons une vaste classe de suites d'entiers naturels "reconnus" par automates. Nous appliquons ensuite ces résultats au problème de décision de la théorie SOM de structures du type $\langle \mathbb{N}, +1, P_1, \ldots, O_n \rangle$, où les P_i sont des prédicats unaires, ainsi qu'au problème de décision de l'égalité de deux suites de nombres rationnels.

Plan détaillé

Nous présentons dans la première partie les définitions de base portant sur les k-piles et les objets les utilisant. Nous introduisons les principales notions qui seront utilisées tout au long de ce manuscrit ainsi que les principaux résultats connus portant sur ces notions.

Le chapitre I.1 est dédié à la définition de *piles itérées*, ainsi que de diverses variantes de cette définition. Nous introduisons également la notion d'*instruction de piles* et définissons les instructions les plus courantes. Nous terminons ce chapitre en donnant une autre représentation des k-piles, sous forme de mots dans un groupe libre. Ce codage sera la base de l'obtention de nombreux résultats de ce manuscrit. Originalement évoqué dans [Mas76], cette représentation est également utilisée dans [Car05].

Le second chapitre présente les automates à k-piles dans le cadre plus général des machines utilisant les k-piles comme mémoire. Nous introduisons également diverses normalisations, ainsi qu'une extension : les *automates à k-piles contrôlés*, consistant à conditionner l'application des instructions par des contrôles sur la pile.

Le dernier chapitre de la partie I introduit les bases de la logique du Second Ordre Monadique (SOM). Nous énumérons les résultats de base portant sur les liens entre la logique SOM et les automates à k-piles.

Dans la seconde partie, nous étudions les ensembles de mots et de piles d'un point de vue logique. Nous définissons une notion étendue de *régularité* liée à la notion de *définissabilité* en LSOM dans une structure arborescente. Initiée par les travaux de Büchi sur les mots, l'étude des liens entre automates et logique s'est avérée être une voie fructueuse en informatique

théorique. Un résultat essentiel obtenu de cette façon par Rabin est la décidabilité de la théorie SOM des arbres infinis. À partir des arbres infinis, il est possible de construire des structures plus complexes ayant une théorie SOM décidable, par l'utilisation de théorèmes de transfert de décidabilité (par exemple le transfert d'une structure à sa *tree-like structure* (voir [MS92] ou [Wal02])). C'est cette approche que nous utilisons ici afin d'étudier la logique SOM de structures d'arbres *étiquetés*. Toutefois, en plus de la décidabilité nous nous intéressons aux ensembles *définissables* en LSOM dans ces structures et aux classes d'automates permettant de les reconnaître.

Le but de cette démarche est de généraliser les différentes façons de caractériser les langages réguliers et particulièrement, l'égalité entre les trois classes suivantes :

- les langages reconnus par automates finis,
- les ensembles de sommets définis par une formule SOM dans une structure arborescente
- les ensembles de 1-piles générés par des automates à piles (en considérant les 1-piles comme des mots).

Nous identifions une extension des automates finis, ainsi que des structures d'arbres étendues par des relation unaires, pour lesquelles les deux premières équivalences ci-dessus sont vérifiée. Nous exhibons ensuite des sous-classes particulières de ces objets permettant de généraliser la troisième équivalence au cas des automates à k-piles.

Cette deuxième partie est découpée en trois chapitres. Nous définissons dans le chapitre II.4 une classe d'automate de mots, utilisant des *oracles* (i.e., ensembles de mots sur l'alphabet d'entrée) pour tester les préfixes déjà calculés des mots en entrée. Les langages reconnus par ces automates sont ensuite reliés à la logique sur des structures d'arbres étiquetés. Cette approche avait été utilisée dans [Tho78] pour caractériser des sous-classes des langages réguliers par l'utilisation de oracles réguliers et pour étudier leur définissabilité en Logique du Premier Ordre sur des structures de mots étendues. Toutefois, la définition d'automates avec oracles testant les préfixes n'apparaît pas explicitement, puisque les oracles réguliers peuvent être simulés par produit direct d'automates finis.

Nous utilisons ensuite ces liens afin de transférer des propriétés à une structure d'arbre, à partir de sa "structure image" par un morphisme. Étant donnés un alphabet fini A, un morphisme de monoïde $\mu : A^* \to M$ surjectif et une structure relationnelle $\mathcal{S}$ sur A^*. La *structure image* $\mu(\mathcal{S})$ de $\mathcal{S}$ a pour domaine M et ses relations sont les images par μ des relations de $\mathcal{S}$. En utilisant la théorie des jeux pour lier logique et automates, nous prouvons deux théorèmes de transferts de $\mu(\mathbf{T}(A)^{\vec{O}})$ vers $\mathbf{T}(A)^{\vec{O}}$ (l'arbre infini de domaine A^* étiqueté par un nombre fini de relations O_i) et procédons à une étude "structurelle" des ensembles SOM-définissables dans $\mathbf{T}(A)^{\vec{O}}$.

Sous l'hypothèse que chaque O_i est l'image inverse par μ d'un sous-ensemble du monoïde M, nous obtenons les principaux résultats suivants :

- *transfert de la décidabilité* (théorème 4.3.17) : si $\mu(\mathbf{T}(A)^{\vec{O}})$ a une SOM-théorie décidable, alors $\mathbf{T}(A)^{\vec{O}}$ a une SOM-théorie décidable.
- *transfert de la propriété du Modèle Définissable* (théorème 4.3.19) : sous une condition sur μ, si $\mu(\mathbf{T}(A)^{\vec{O}})$ vérifie la propriété du *Modèle Définissable* (MD), alors $\mathbf{T}(A)^{\vec{O}}$ vérifie également MD. Cette propriété assure à une structure $\mathcal{S}$ que toute formule satisfiable admet au moins un modèle SOM-définissable dans $\mathcal{S}$ (voir définition 3.1.2).
- *structure des définissables* (théorème 4.3.20) : sous la même condition sur μ, et si $\mu(\mathbf{T}(A)^{\vec{O}})$ satisfait MD, alors un ensemble est SOM-définissable dans $\mathbf{T}(A)^{\vec{O}}$ ssi il est

reconnu par un automate fini utilisant uniquement des oracles de la forme $\mu^{-1}(D)$ où D est SOM-définissable dans $\mu(\mathbf{T}(A)^{\vec{O}})$. (Ainsi chaque oracle teste une propriété SOM-définissable dans $\mu(\mathbf{T}(A)^{\vec{O}})$, sur l'image par μ des préfixes des mots en entrée).

Il existe différentes classes de prédicats unaires P conduisant à une structure $\langle \mathbb{N}, +1, P\rangle$ dont la théorie SOM est décidable (voir [ER66, CT02, FS03], ainsi que la partie VI). Une application immédiate du premier théorème de transfert est que pour un tel prédicat, la structure $\mathbf{T}(A)^{|P|^{-1}}$ admet une SOM-théorie décidable (où $|P|^{-1}$ est l'ensemble des mots sur A^* dont la longueur satisfait P)

En appliquant les théorèmes ci-dessus, nous obtenons des classes de langages admettant deux caractérisations équivalentes : comme langages reconnus par automates à oracles, et comme ensembles SOM-définissables dans une structure d'arbre étiqueté. Ces caractérisations généralisent celles des langages réguliers.

Nous étendons finalement la troisième caractérisation des langages réguliers (comme ensembles de mots de piles) aux piles itérées. Dans le chapitre II.5, nous appliquons itérativement les trois théorèmes sur une famille de structures $(\boldsymbol{\mathcal{P}}_{\mathbf{k}})_{k\geq 1}$ ayant pour domaine le langage de mots $\mathcal{P}_k$. Ce langage correspond au codage des k-piles défini dans la partie I.

Dans le chapitre II.6, nous appliquons ces résultats aux piles itérées. La structure $\boldsymbol{\mathcal{P}}_{\mathbf{k}}$ est SOM-*équivalente* à la structure $\mathbf{Pile}_{\mathbf{k}}$ dont le domaine est l'ensemble des k-piles et dont les relations sont celles induites par les instructions de piles push, pop et change. Ceci permet de définir une classe de langages dans $\mathcal{P}_k$ pouvant être exprimée de trois manière équivalentes (théorème 6.2.7) :

- comme langages reconnus par automates finis avec oracles
- comme codage d'ensembles de k-piles définis par SOM-formules dans $\mathbf{Pile}_{\mathbf{k}}$
- comme codage des ensembles de k-piles générés par automate de niveau k contrôlé.

Nous prouvons de plus que $\mathbf{Pile}_{\mathbf{k}}$ satisfait la propriété du Modèle Définissable et admet une SOM-théorie décidable (théorème 6.2.1). Le résultat portant sur la décidabilité avait déjà été prouvé dans [FS03] en utilisant le théorème de transfert prouvé dans [Wal02].

Une notion de "régularité" sur les ensembles de piles itérées à également été définie dans [Car05], par le biais d'une forme normale générale. En utilisant les liens forts (voir [CW03]) existant entre la hiérarchie de graphes dite "de Caucal" ([Cau96]) et la classe des graphes de calculs des automates à k-piles, il y est également donné une preuve de l'égalité entre les ensembles définissables de k-piles, et ceux répondant à cette normalisation. Les liens entre ces deux travaux disjoints mettent en évidence le fait que la notion choisie d'ensemble "réguliers" de piles itérées semble être la plus adéquate à la généralisation de la notion d'ensembles réguliers.

Ces résultats sont utilisés dans la partie III, pour se recentrer sur la théorie des langages et des automates. Cette partie vise essentiellement à identifier clairement les différentes classes d'automates permettant de reconnaître les langages de niveau k . Elle consiste donc en comparaison de classes d'automates et en applications de ces résultats. Nous introduisons pour cela la notion de *simulation déterministe* d'un automate par un autre. Cette notion, présentée dans le chapitre III.7, permet de préserver de nombreuses propriétés des automates. En particulier, deux classes d'automates en relation par simulation déterministe reconnaissent la même classe de langage, et admettent la même classe de langages **déterministes**. Toutes les équivalences prouvées dans cette partie sont des équivalences pour cette relation.

Le chapitre III.8 est consacré à la preuve de l'équivalence entre les automates à k-piles et différentes restrictions de cette classe. En particulier, une restriction courante consiste à utiliser des automates dont les alphabets de piles de niveaux supérieurs à 1 sont réduits à un seul élément. Il est communément admis que ces automates reconnaissent également les langages de niveau k, toutefois, à notre connaissance, aucune preuve complète de ce résultat n'a jamais été publiée. Nous établissons que cette classe d'automate est équivalente, par simulation déterministe, à celle des automates à k-piles, et étendons ce résultat au cas des automates *contrôlés*. Nous prouvons également que la classe d'équivalence est conservée si on supprime l'utilisation de l'instruction change consistant à échanger un symbole en tête de pile. Tous ces résultats sont résumés dans le théorème 8.2.4.

Nous étudions dans le chapitre III.9 l'apport de l'utilisation de contrôleurs sur les piles. Les contrôles permettent de restreindre l'application des transitions selon l'appartenance de la pile courante à un nombre fini d'ensembles de piles itérées. Nous prouvons principalement l'équivalence entre les automates à k-piles non contrôlés et les classes d'automates suivantes :

- automates à k-piles contrôles par des ensembles "réguliers" de k-piles, c'est-à-dire, SOM-définissables dans la structure $\mathbf{Pile_k}$ associée au type de donnée "k-piles" (théorème 9.3.7)
- automates à instructions "symétriques", c'est-à-dire tels que les instructions pop_i sont remplacées par des instruction $\overline{\mathrm{push}}_{a_i}$ symétriques des instruction push_{a_i} : le dépilement n'est autorisé que si la tête de pile de niveau i est a_i et les deux piles de niveau $i-1$ au sommet de pile sont identiques (théorème 9.3.8).
- automates à 1-piles contrôlés par des ensembles de mots tous reconnaissables par un même automate à $(k-1)$-piles **déterministe** (pour des états finaux différents) (théorème 9.3.11).

Le dernier chapitre est constitué d'applications de ces résultats. Nous décrivons tout d'abord diverses applications en théorie des langages, à la résolution de problèmes de reconnaissabilité et de décision. Nous donnons également une nouvelle caractérisation des langages de niveau k (théorème 10.1.3), comme projection de langages *k-réguliers* (i.e., de codages sous forme de mots d'ensembles "réguliers" de k-piles).

Nous exhibons ensuite un générateur du cône rationnel principal formé par la classe des langages de niveau k. Enfin, nous étudions les *jeux de parité* joués dans le graphe de calcul d'un automate à piles itérées. Nous prouvons qu'il est possible de décider du gagnant d'un tel jeu et qu'il existe une stratégie pour le gagnant calculée par un automate à piles itérées. Ces derniers résultats ont déjà été établis de manière efficace dans [Cac03].

Dans la dernière partie, nous développons une approche de l'étude de suites de nombres par le biais d'automates à k-piles.

Nous définissons tout d'abord la classe $\mathbb{S}_k$ des suites "calculées" par automates à k-piles déterministe. Nous donnons un grand nombre de familles de suites ainsi calculées (par exemple, le niveau 2 contient les suites *rationnelles* d'entiers (voir [BR88])), et prouvons que la classe $\mathbb{S}_k$ est close par de nombreuses opérations élémentaires, ainsi que par des opérations plus complexes telles que la résolution de systèmes d'équations polynomiales à coefficients dans des classes de suites (le théorème 11.5.19 résume ces propriétés de clôture). Nous fournissons ainsi un grand nombre d'exemples de langages de niveau k.

Nous appliquons ensuite ces résultats à la résolutions de problèmes d'arithmétiques. Nous définissons une hiérarchie de classes de suites d'entiers, construite à partir de $(\mathbb{S}_k)_{k\geq 0}$, et bénéficiant de la plupart de ses propriétés de clôture (théorème 13.2.14). Nous prouvons que

pour toute suite s de cette nouvelle classe, la théorie du Second Ordre Monadique de la structure $\langle \mathbb{N}, +1, s(\mathbb{N}) \rangle$ est décidable (théorème 13.1.6). Cette classe contient par exemple les suites $(n\lfloor\sqrt{n}\rfloor)_{n\in\mathbb{N}}$ et $(n\lfloor\log n\rfloor)_{n\in\mathbb{N}}$. Nous apportons ainsi une nouvelle méthode pour la résolution des problèmes de ce type, déjà étudiés par exemple dans [ER66, Sie70, Sem84, Mae99, CT02]. Cette résultats se généralisent également au cas de structures admettant plusieurs prédicats emboîtés comme par exemple $\langle \mathbb{N}, +1, \{n^{k_1}\}_{n\geq 0}, \{n^{k_1 k_2}\}_{n\geq 0}, \ldots, \{n^{k_1 \cdots k_m}\}_{n\geq 0} \rangle$ où les k_i sont des entiers positifs.

Nous achevons ce manuscrit en introduisant une large classe $\mathcal{F}(\mathbb{S}_k)$ de suites de nombres **rationnels**, également close par de nombreuses opérations. Le niveau 3, par exemple, contient les suites de rationnels dites "P-récurrentes", correspondant aux séries formelles D-finies (voir [Sta80] pour un survol). Nous prouvons que pour les suites dans $\mathcal{F}(\mathbb{S}_k)$, le problème de savoir si deux suites sont égales se réduit au problème de l'équivalence de deux automates à k-piles **déterministes**.

Ceci établit un pont entre les problèmes algorithmiques sur les suites (traités dans [PWZ96], par exemple) et les problèmes de décision sur les automates (traités dans [Sén02], par exemple).

Table des matières

Préliminaires

Cette partie introduit les notations et définitions de base qui seront fréquemment utilisées au cours de ce document.

0.1 Mots

Monoïde libre

Étant donné un ensemble fini A (appelé **alphabet**), un **mot** sur A est une séquence finie et possiblement vide $a_1 \cdots a_n$, de lettres de A. Un **langage** est un ensemble de mots. Le langages contenant tous les mots sur l'alphabets A est noté A^*. L'ensemble A^* peut-être muni d'une loi interne appelée **produit de concaténation** lui conférant une structure de monoïde libre. Si $u = a_1 \cdots a_n$ et $v = b_1 \cdots b_m$ sont deux mots de A^*, la concaténation de u et v est notée $u \cdot v$ (ou simplement uv) et désigne le mot $a_1 \cdots a_n b_1 \cdots b_m$. Le mot vide est noté ε et correspond à l'élément neutre de ce monoïde.

Pour un mot donné $u \in A^*$, la longueur de u est désignée par $|u|$, et pour tout $n \geq 0$, l'ensemble des mots sur A de longueur n est A^n. Pour deux mots u et v, nous notons $u \preccurlyeq v$ si u est un préfixe de v, c'est à dire si il existe un mot w tel que $uw = v$. L'image miroir d'un mot $u = a_1 \cdots a_n$ est $\tilde{u} = a_n \cdots a_1$.

Groupe libre

Afin d'obtenir une structure de groupe sur les mots, nous associons à chaque lettre a d'un alphabet A, un symbole inverse noté $\bar{a}$ et n'appartenant pas à A. Notons $\overline{A}$ l'ensemble des éléments inverses de A et $\widehat{A}$ l'alphabet formé de A et de ses lettres inverses : $\widehat{A} = A \cup \overline{A}$.

Nous étendons maintenant la notion d'inverse aux mots. L'inverse d'un mot $u \in \widehat{A}^*$ est noté $\overline{u}$ et est obtenu en transformant chaque lettre de u en son inverse, puis en prenant l'image miroir de ce mot. Formellement, si $u = a_1 \cdots a_n \in \widehat{A}^*$, alors $\overline{u} = b_n \cdots b_1$ où pour tout $i \in [1, n]$:

- si $a_i \in A$, alors $b_i = \bar{a}_i$,
- si $a_i = \bar{a} \in \overline{A}$, alors $b_i = a$.

L'ensemble $\widehat{A}^*$ peut être muni d'une loi interne lui conférant une structure de groupe. Pour cela, considérons le système semi-thuien $S = \{(a\bar{a}, \varepsilon), (\bar{a}a, \varepsilon)\}$.

Nous dirons qu'un mot de $\widehat{A}^*$ est **réduit**, si il est S-réduit (i.e., on ne peut plus lui appliquer aucune réduction de S). Comme S est confluent et noetherien, chaque mot $w \in \widehat{A}$ est équivalent (mod $\leftrightarrow_S^*$) à un unique mot réduit que nous noterons $\rho(w)$. Le groupe libre $(F(A), 1, \bullet)$ est le quotient $\widehat{A}^* / \leftrightarrow_S^*$. Nous désignons par $\mathrm{Irr}(A)$ l'ensemble des mots réduits de $\widehat{A}^*$. Les mots composants $\mathrm{Irr}(A)$ ne possèdent donc aucune occurence de la forme $a\bar{a}$ ou $\bar{a}a$.

Le plus souvent, nous ne considérons que les représentants réduits de $F(A)$ et nous utiliserons donc indifféremment les éléments de $F(A)$ et ceux de $\mathrm{Irr}(A)$. De même, le produit $\bullet$ est étendu implicitement aux éléments de $\mathrm{Irr}(A)$ et ($\mathrm{Irr}(A)$, $\bullet$, ε) forme donc un groupe. Si u et v appartiennent à $\mathrm{Irr}(A)$, le produit $u \bullet v$ correspond à la réduction de la concaténation de u avec v : $u \bullet v = \rho(uv)$.

0.2 Fonctions

Extensions

Soit $f : E \to F$ une fonction quelconque. Nous étendrons systématiquement (et implicitement) f aux éléments des différents types suivants :

- aux sous-ensembles de E : si $S \subseteq E$, alors $f(S) = \{f(e) \mid e \in S\}$
- aux ensembles de sous-ensembles de E : si P est inclus dans les parties de E, alors $f(P) = \{f(S) \mid S \in P\}$
- aux vecteurs d'éléments de E : si $\vec{v} = (e_1, \ldots, e_n)$, avec chaque $e_i \in E$, alors $f(\vec{v}) = (f(e_1), \ldots, f(e_n))$,
- aux vecteurs de sous-ensembles de E : si $\vec{V} = (S_1, \ldots, S_n)$, et chaque $S_i \subseteq E$, alors $f(\vec{V}) = (f(S_1), \ldots, f(S_n))$,
- aux ensembles de vecteurs de sous-ensembles de E : $f(P) = \{f(\vec{V}) \mid \vec{V} \in P\}$,

Projections

Pour tous entiers i, j, n, avec $0 \leq i \leq j \leq n$, et tout vecteur d'éléments $(a_1, \ldots, a_n)$, nous notons

$$\pi_i(a_1, \ldots, a_n) = a_i \text{ et } \pi_{i \cdots j}(a_1, \ldots, a_n) = (a_i, \ldots, a_j)$$

Étant donnés deux alphabets B et A avec $B \subseteq A$, la projection $\pi_B : A^* \to B^*$ désigne le morphisme consistant à supprimer d'un mot toutes les lettres qui n'appartiennent pas à B : $\forall a \in A$, $\pi_B(a) = a$ si $a \in B$ et $\pi_B(a) = \varepsilon$ sinon.

Vecteur caractéristique

Soit E un ensemble et $\vec{S} = (S_1, \ldots, S_n)$ un vecteur de sous-ensembles de E. Pour tout $e \in E$, le vecteur caractéristique de e dans $\vec{S}$ est un vecteur booléen $\chi_{\vec{S}}(e)$ de taille n indiquant l'appartenance de e à chacun des S_i : $\chi_{\vec{S}}(e) = (b_1, \ldots, b_n)$ où $\forall i$, $b_i = 1$ ssi $e \in S_i$.

0.3 Graphes

Un graphe (orienté) est la donnée d'un ensemble V de sommets et d'un ensemble $E \subseteq V \times V$ d'arcs liant ces sommets :

$$\mathcal{G} = (V, E)$$

Les arcs d'un graphes peuvent être étiquetés par les éléments d'un ensemble Σ. Dans ce cas, la relation E est indicée par les éléments de Σ. Un graphe étiqueté sur Σ est donc la donnée d'un ensemble V de sommets et d'une famille indicée par Σ, de relations binaires sur V :

$$\mathcal{G} = (V, (E_\alpha)_{\alpha \in \Sigma})$$

0.4 Arbres et forêts

Soient Σ et A deux alphabets finis et $P \subseteq A^*$ un langage clos par préfixe (i.e., si P contient un mot u, alors il contient également tous ses préfixes). Un arbre de domaine P étiqueté par

Σ est une fonction totale $t : P \to \Sigma$. Une foret est une ensemble d'arbres. La forêt contenant tous les arbres de domaine P étiquetés sur Σ est noté P-Arbre(Σ).

Nous utiliserons deux types d'opérations sur les arbres et forêts :

Restriction : La restriction d'un arbre $t \in A^*$-Arbre(Σ), à un domaine P clos par préfixe est un P-arbre(Σ) noté $t_{|P}$.

Cette restriction s'applique également aux forêts. Si $F \subseteq A^*$-Arbre(Σ), alors $F_{|P}$ est l'ensemble des arbres $t_{|P}$ pour $t \in F$.

Produit : Soient $t_1 \in P$-Arbre(Σ_1) et $t_2 \in P$-Arbre(Σ_2), le produit (cartésien) de t_1 et t_2 est l'arbre $t_1 \widehat{} t_2$ de P-Arbre($\Sigma_1 \times \Sigma_2$) vérifiant $\forall u \in P$,

$$t_1 \widehat{} t_2(u) = (t_1(u), t_2(u))$$

Cette définition se généralise naturellement aux forêts :
si F_1, $F_2 \subseteq P$-Arbre(Σ), alors $F_1 \widehat{} F_2 = \{t_1 \widehat{} t_2 |\ t_1 \in F_1,\ t_2 \in F_2\}$.

Arbre caractéristique

Étant donné un alphabet fini A et $\vec{O} = (O_1, \ldots, O_m)$, $O_i \subseteq A^*$, l'arbre caractéristique de $\vec{O}$ est le A^*-arbre($\{0,1\}^m$) noté $\mathrm{T}(A)^{\vec{O}}$ et défini $\forall u \in A^*$ par $\mathrm{T}(A)^{\vec{O}}(u) = \chi_{\vec{O}}(u)$.

Première partie

Structures à piles itérées

Chapitre 1

Piles k-itérées

Les *piles itérées* sont des structures de mémoire construites itérativement. Originellement définies par [Gre70] dans le cadre général de *Familles abstraites d'automates*, ces structures ont déjà été étudiées sous diverses versions et par de nombreux auteurs. Bien que la plupart des définitions récemment étudiés soient restrictives, nous choisissons ici la description la plus générale, que nous comparerons en détail aux autres définitions.

Nous débutons ce chapitre en fixant la notion de pile k-itérée (où k-pile) et en définissant les outils qui nous permettrons de manipuler ces objets. Les autres versions existantes de structures de mémoire itérées sont ensuite introduites comme des restrictions, à isomorphisme près, de notre notion de k-pile. Nous définissons ensuite un type particulier de fonctions s'appliquant aux k-piles : les *instructions* de piles. Nous présentons les instructions habituellement utilisées, ainsi que d'autres instructions moins communes mais se prêtant assez naturellement à l'étude et à la manipulation des k-piles. Pour terminer, nous donnons une représentation linéaire des k-piles sous forme de mots dans un groupe libre.

1.1 Définitions - Représentation

Une classe $\mathfrak{A}$ d'alphabets de piles est un ensemble au plus dénombrable de suites $(A_k)_{k\geq 0}$ d'alphabets vérifiant

- $A_0 = \emptyset$
- $\forall i \neq j$, les ensembles A_i et A_j sont disjoints.

Pour tout ce manuscrit, nous supposons fixée une classe $\mathfrak{A}$ d'alphabets de piles suffisamment grande pour contenir tous les alphabets dont nous aurons besoin. Toute suite suite $(A_k)_{k\geq 0}$ d'alphabets de piles que nous utiliserons sera considérée implicitement comme appartenant à $\mathfrak{A}$.

Soit $(A_k)_{k\geq 0}$ une suite d'alphabets de piles. Pour tout $k \geq 0$, l'ensemble k-$\mathsf{Pile}(A_1, \ldots, A_k)$ des **piles k-itérées** sur $(A_k)_{k\geq 0}$ est définie inductivement par :

$$\begin{aligned} 0\text{-}\mathsf{Pile} &= \{\bot\} \\ (k+1)\text{-}\mathsf{Pile}(A_1, \ldots, A_{k+1}) &= (A_{k+1}[k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)])^* \bot [k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)], \end{aligned}$$

où $\bot$ est le **symbole de fond de pile** et n'appartient à aucun des A_i.

Par exemple $a_2[a_1[\bot]b_1[\bot] \bot [\bot]]b_2[b_1[\bot] \bot [\bot]] \bot [\bot [\bot]]$ est une 2-pile.

Notations

1. La classe des piles itérées est noté it-Pile et désigne l'union pour $(A_\ell)_{\ell\geq 0} \in \mathfrak{A}$ et $k \geq 0$ de tous les k-Pile$(A_1, \ldots, A_k)$. Si $(A_\ell)_{\ell\geq 0}$ est fixé, it-Pile$(A_1, \ldots, A_k, \ldots)$ est la restriction de it-Pile$(\mathfrak{A})$ aux piles définies sur $(A_\ell)_{\ell\geq 0}$.
2. Nous appelons k-atome tout élément $a[\omega]$ où $\omega \in (k-1)$-Pile et $a \in A_k \cup \{\bot\}$.
3. Nous notons $A_k^\bot$ l'ensemble $A_k \cup \{\bot\}$.
4. Le symbole de fond de pile peut être n'importe quelle lettre n'appartenant pas aux alphabets de piles. Lorsqu'il est nécessaire de préciser le symbole utilisé, nous notons k-Pile$^\bot(A_1, \ldots, A_k)$ (si le symbole est $\bot$).
5. La **pile vide** de niveau k, est notée $\bot_k$ et correspond à la k-pile ne contenant que le symbole $\bot$:
   $\bot_0 = \bot$ et $\bot_{k+1} = \bot\,[\bot_k]$.
6. L'ensemble des piles itérées sur les alphabets $(A_k)_{k\geq 0}$ est noté it-Pile$(A_1, \ldots, A_k, \ldots)$ et désigne l'union de tous les k-Pile$(A_1, \ldots, A_k)$, pour $k \geq 0$.
7. Toute 1-Pile est de la forme $a_1[\bot]a_2[\bot]\cdots a_n[\bot]\,\bot\,[\bot]$, $n \geq 0$. Toutefois, nous représenterons (abusivement), une telle 1-pile par le mot $a_1 \cdots a_n\,\bot$. Nous appliquerons également cette simplification aux 1-piles apparaissant au sein d'une k-pile (voir exemple ci-après).
8. Par définition, tout ω dans $(k+1)$-Pile, $k \geq 0$, admet une unique décomposition de la forme
   $$\omega = a[\omega_1]\omega'$$
   où $\omega_1 \in k$-Pile, $\omega' \in (k+1)$-Pile $\cup \{\varepsilon\}$, $a \in A_{k+1} \cup \{\bot\}$ et $[a = \bot$ ssi $\omega' = \varepsilon]$.

Exemple 1.1.1. *Soit* $A_1 = \{a_1\}$, $A_2 = \{a_2, b_2\}$, $A_3 = \{a_3\}$,
$\omega_{ex} = a_3[a_2[a_1a_1\,\bot]b_2[a_1\,\bot]\,\bot_2]\,\bot\,[a_2[a_1\,\bot]a_2[\bot]\,\bot_2] \in 3\text{-Pile}(A_1, A_2, A_3)$.
Sa décomposition est $\omega_{ex} = a[\omega_1]\omega'$ *avec*
$a = a_3$, $\omega_1 = a_2[a_1a_1\,\bot]b_2[a_1\,\bot]\,\bot_2$ *et* $\omega' = \bot\,[a_2[a_1\,\bot]a_2[\bot]\,\bot_2]$.

Toute k-pile peut-être représentée par une suite d'arbres planaires complets de hauteur k et dont les noeuds de profondeur i sont étiquetés par des lettres de $A_i \cup \{\bot\}$.

La figure 8 illustre la représentation planaire de la 3-pile donnée dans l'exemple 1.1.1.

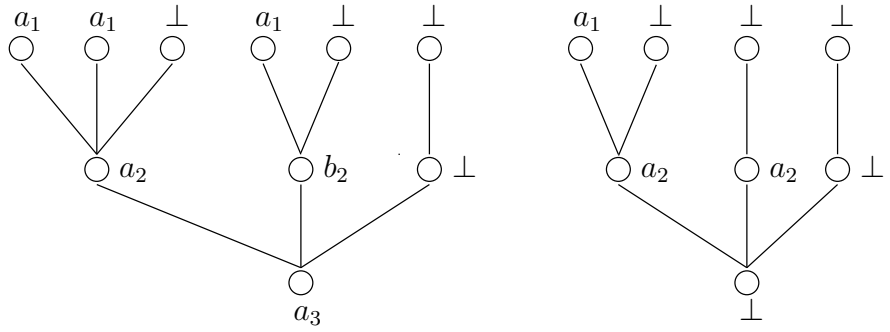


FIG. 8 – Représentation planaire de la 3-pile ω_{ex}

Trois applications seront fréquemment utilisées pour la description et la manipulation des piles itérées :

- la **projection** p_i, associant à toute pile ω, l'élément de i-Pile du sommet de ω.
- la fonction de lecture mot_i, associant à toute it-pile ω, **l'image miroir** du mot de A_i^* apparaissant dans **projection** de ω dans i-Pile
- la fonction top, lisant la **liste des symboles en tête** de chaque niveau.

Définition 1.1.2 (Projection). *L'application projetant une k-pile sur sa i-pile de sommet de pile est* $\mathrm{p}_{k,i}$ *:* k-Pile$(A_1, \ldots, A_k) \to i$-Pile$(A_1, \ldots, A_i)$, $0 \leq i \leq k$, *et vérifie* $\forall \omega = a[\omega_1]\omega' \in k$-Pile *:*

- $\mathrm{p}_{k,k}(\omega) = \omega$,
- $\mathrm{p}_{k,i}(\omega) = \mathrm{p}_{k-1,i}(\omega_1)$ *si* $1 \leq i \leq k-1$.

Plus généralement, la fonction $\mathrm{p}_i : it$-Pile $\to i$-Pile *est définie pour tout* $\omega \in k$-Pile, $k \geq 0$ *:*

$\mathrm{p}_i(\omega) = \mathrm{p}_{k,i}(\omega)$ *si* $k \geq i$ *et* $\mathrm{p}_i(\omega)$ *est indéfini sinon.*

Sur la représentation planaire de ω, $\mathrm{p}_i(\omega)$ correspond à la suite des sous-arbres du noeud de niveau $i+1$ le plus à gauche.

Exemple 1.1.3. *Soit* ω_{ex} *la 3-pile donnée exemple 1.1.1 :*

$\mathrm{p}_3(\omega_{ex}) = \omega_{ex}$, $\mathrm{p}_2(\omega_{ex}) = a_2[a_1a_1 \perp]b_2[a_1 \perp] \perp_2$ *et* $\mathrm{p}_1(\omega_{ex}) = a_1a_1 \perp$.

Définition 1.1.4 (Mot de niveau i). *La fonction associant à toute k-pile, l'image miroir du mot de A_i^* apparaissant dans la proction au niveau i de la pile est* $\mathrm{mot}_{k,i} : k$-Pile$(A_1, \ldots, A_k) \to A_i^*$, $0 \leq i \leq k$ *et vérifie* $\forall \omega = a_n[\omega_n] \cdots a_1[\omega_1] \perp [\omega_0] \in k$-Pile, $n \geq 0$ *:*

- $\mathrm{mot}_{k,k}(\omega) = a_1 \cdots a_n$
- $\mathrm{mot}_{k,i}(\omega) = \mathrm{mot}_{k-1,i}(\omega_n)$ *si* $1 \leq i \leq k-1$.

Plus généralement, définissons l'application $\mathrm{mot}_i : it$-Pile$(A_1, \ldots, A_k, \ldots) \to A_i^*$, $i \geq 1$, *telle que pour tout* $\omega \in k$-Pile, $k \geq 0$ *:*

$\mathrm{mot}_i(\omega) = \mathrm{mot}_{k,i}(\omega)$ *si* $k \geq i$ *et* $\mathrm{mot}_i(\omega) = \varepsilon$ *sinon.*

Sur la représentation planaire de ω, $\mathrm{mot}_i(\omega)$ correspond à la lecture de droite à gauche, des étiquettes des fils du noeud de niveau $i+1$ le plus à gauche.

Exemple 1.1.5. *Soit* ω_{ex} *la 3-pile donnée dans l'exemple 1.1.1.*

$\mathrm{mot}_3(\omega_{ex}) = a_3$, $\mathrm{mot}_2(\omega_{ex}) = b_2a_2$, $\mathrm{mot}_1(\omega_{ex}) = a_1a_1$.

La figure 9 montre les résultats obtenus en appliquant les fonctions p_i et mot_i à la 3 repésentée par la figure 8.

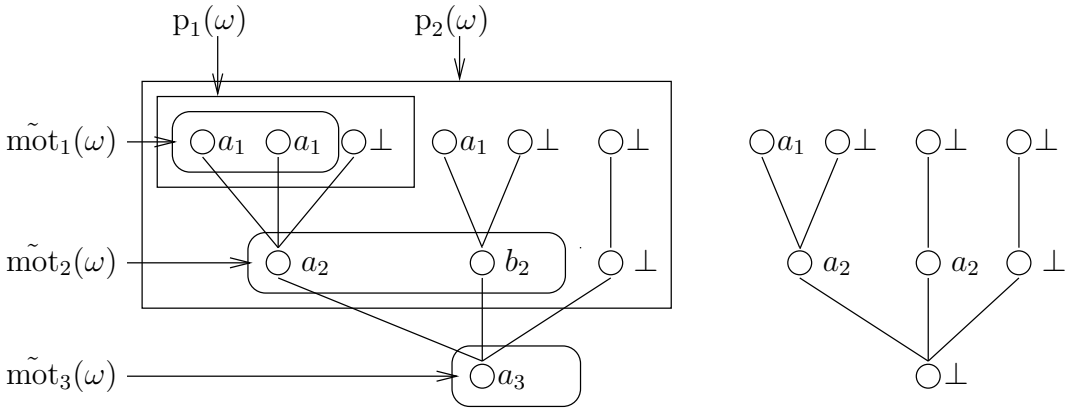


FIG. 9 – Les fonctions mot_i et p_i appliquées à ω_{ex}

Remarque 1.1.6. *Dans la définition de* mot_i*, nous choisissons de considérer l'image miroir du mot de pile de niveau* i *pour représenter facilement le résultat de l'empilement d'un mot au niveau* i*. Si* $u = a_1 \cdots a_n$*, alors l'application successive des instructions* $\mathrm{push}_{i,a_1}, \mathrm{push}_{i,a_2}, \ldots, \mathrm{push}_{i,a_n}$ *à une pile* ω *donne une pile* ω' *vérifiant : le prefixe de* $\mathrm{mot}_i(\omega')$ *est* u*. Rapellons que traditionellement, un automate lit les mots en entrée de la droite vers la gauche.*

Définition 1.1.7 (Symboles de tête). *L'application associant à toute* k*-pile,* $k \geq 1$*, la liste de ses* k *symboles de tête est* $\mathrm{top} : k\text{-Pile}(A_1, \ldots, A_k) \to {A_k}^{\perp} \cdots {A_1}^{\perp}$*, définie pour tous* $\omega_1 \in (k-1)\text{-Pile}$*,* $\omega \in k\text{-Pile} \cup \{\varepsilon\}$ *et* $a \in A_k \cup \{\perp\}$ *par*

- $\mathrm{top}(a\omega) = a$*, si* $k = 1$
- $\mathrm{top}(a[\omega_1]\omega) = a \cdot \mathrm{top}(\omega_1)$*, si* $k \geq 2$.

Sur la représentation planaire de ω, $\mathrm{top}_i(\omega)$ correspond à la lecture, de bas en haut des l'étiquettes les plus à gauche de chaque niveau. Nous notons $\forall 1 \leq i \leq j \leq k$, $\mathrm{top}_i(\omega) = a_i$ et $\mathrm{top}_{i\cdots j}(\omega) = a_i \cdots a_j$.

Exemple 1.1.8. *Soit* ω_{ex} *la 3-pile donnée dans l'exemple 1.1.1 :*
$\mathrm{top}(\omega_{ex}) = a_3a_2a_1$, $\mathrm{top}_3(\omega_{ex}) = a_3$, $\mathrm{top}_2(\omega_{ex}) = a_2$ *et* $\mathrm{top}_1(\omega_{ex}) = a_1$.

1.2 Quelques variantes

Nous présentons ici les variantes existantes de la notion de structures de mémoire itérées. Toutes ces notions admettent une représentation planaire sous forme de suite d'arbres mais différent dans la forme des arbres ou dans la manière de les étiqueter. Ces différentes définitions correspondent (à isomorphisme près) à des sous-classes remarquables de l'ensemble des piles itérées.

1.2.1 Piles sans fond de piles

Une première version légèrement plus simple consiste à utiliser le même type de structures que k-Pile mais sans utiliser de fond de piles (voir [Mas76, DG86, FS03]). Au niveau k, de tels objets peuvent donc être représenté par des suites finis d'arbres de hauteur k mais non nécessairement complets. L'utilisation de fonds de piles permet l'existence de piles dont les niveaux supérieurs sont vides, mais pas les niveaux inférieurs, comme par exemple $\perp\ [a_1b_1\ \perp]$.

Considérons la restriction k-NPile de k-Pile composé de toutes les k-piles telles que chaque occurrence du symbole $\perp$ au niveau i, fait partie d'une occurrence de $\perp_i$. L'ensemble k-NPile est défini pour toute suite d'alphabets de piles $(A_\ell)_{\ell \geq 0}$, par

- $0\text{-NPile} = \perp$
- $(k+1)\text{-NPile}(A_1, \ldots, A_{k+1}) = (A_{k+1}[k\text{-NPile}(A_1, \ldots, A_k)])^*\ \perp_{k+1}$

Par exemple $\omega = a_3[\perp_2]b_3[a_2[a_1\ \perp]\ \perp_2]\ \perp_3$ est une 3-Npile mais l'exemple donné ci-dessus n'en est pas une.

La classe k-NPile est isomorphe à celles étudiées dans [Mas76, DG86, FS03] et décrit une restriction très importante de k-Pile car elle peut être munie d'une structure de monoïde libre engendré par l'alphabet infini formé de tous les k-atomes. En effet, la position fixe des symboles $\perp$ permet de définir un produit de *concaténation* sur k-NPile : si $\omega = \omega_1\ \perp_k$ et $\omega' = {\omega_1}'\ \perp_k$ sont deux éléments de k-NPile, alors le produit $\omega \cdot \omega'$ désigne la pile $\omega_1 \cdot {\omega_1}' \cdot\ \perp_k$ appartenant à k-NPile. Remarquons que $1\text{-Pile}(A_1) = 1\text{-NPile}(A_1)$ et peut donc toujours être muni d'un produit de concaténation.

1.2.2 Piles "d'ordre supérieur"

La définition la plus restrictive, et en même temps, la plus utilisée récemment (voir par exemple [KNU02, Cac03, CW03]) est celle de "piles d'ordre supérieur" (POS). Une 1-POS sur un alphabet A est une liste $[a_1, \ldots, a_k]$ d'éléments de A et une $(k+1)$-POS est une liste $[\omega_1, \ldots, \omega_k]$ de k-POS. Toute pile dans k-POS peut donc être représentée par une suite d'arbres planaires de hauteur k, complets, et étiquetés uniquement aux feuilles.

La classe k-POS(A) n'est donc pas incluse dans k-Pile, mais est isomorphe à toute classe k-Pile$(A, A_2, \ldots, A_k)$, pour $A_2, \ldots, A_k$ réduits à un unique élément.

1.3 Instructions de piles

Soit $(A_k)_{k\geq 0}$ un alphabet de piles. Une **instruction** sur $(A_k)_{k\geq 0}$ est une fonction (partielle) de it-Pile$((A_k)_{k\geq 0})$ dans it-Pile$((A_k)_{k\geq 0})$ telle que l'application de l'instruction à une pile laisse stable son niveau : pour tout $\omega \in k$-Pile$(A_1, \ldots, A_k)$, $k \geq 0$, instr$(\omega) \in k$-Pile$(A_1, \ldots, A_k)$. L'instruction la plus simple est l'instruction stay définie pour tout $\omega \in it$-Pile par stay$(\omega) = \omega$.

Une **instruction de niveau** i (ou i-instruction) est une instruction qui ne modifie pas les niveaux supérieurs à i des piles auxquelles elle est appliquée. Si instr est une i-instruction, alors :

- pour tout $\omega = a[\omega_1]\omega' \in k$-Pile avec $k > i$, instr$(\omega) = a[\text{instr}(\omega_1)]\omega'$
- pour tout $\omega \in k$-Pile avec $k < i$, instr$(w) = w$.

Une k-instruction est donc complètement définie par sa restriction à k-Pile.

1.3.1 Classes d'instructions

Nous formalisons maintenant la notion de *classes d'instructions* permettant d'obtenir des familles "indépendantes" des alphabets de piles choisis. Définissons tout d'abord la notion d'application compatible. Soient $(A_k)_{k\geq 0}$ et $(B_k)_{k\geq 0}$ des alphabets de piles. Une application **compatible** de $(A_k)_{k\geq 0}$ dans $(B_k)_{k\geq 0}$ est une application $\beta : \bigcup_{k\geq 0} A_k \to \bigcup_{k\geq 0} B_k$ laissant stables les niveaux des alphabets : pour tout $k \geq 0$, $\beta(A_k) \subseteq B_k$.
Une telle application peut-être étendue aux piles en $\beta : it$-Pile$((A_k)_{k\geq 0}) \to it$-Pile$((B_k)_{k\geq 0})$: de la façon suivante : pour tout $\omega \in k$-Pile$(A_1, \ldots, A_k)$,

- si $\omega = \perp_k$ alors $\beta(\omega) = \perp_k$
- si $\omega = a[\omega_1]\omega'$, alors $\beta(\omega) = \beta(a)[\beta(\omega_1)]\beta(\omega')$

Une **classe d'instructions** est une famille $\mathfrak{I}$ d'instructions de piles (définies sur les suites alphabets de la classe $\mathfrak{A}$ fixée au début de ce chapitre) et vérifiant les propriétés suivantes :

- pour tout $(A_k)_{k\geq 0} \in \mathfrak{A}$, l'instruction stay : it-Pile$((A_k)_{k\geq 0}) \to it$-Pile$((A_k)_{k\geq 0})$ appartient à $\mathfrak{I}$,
- pour tous alphabets $(A_k)_{k\geq 0}$ et $(B_k)_{k\geq 0}$ dans $\mathfrak{A}$ et toute **injection compatible** β : $(A_k)_{k\geq 0} \to (B_k)_{k\geq 0}$, pour toute instruction instr $\in \mathfrak{I}$ sur $(A_k)_{k\geq 0}$, il existe instr$_\beta \in \mathfrak{I}$, instruction sur $(B_k)_{k\geq 0}$ telle que $\forall \omega \in it$-Pile$((A_k)_{k\geq 0})$, $\beta(\text{instr}(\omega)) = \text{instr}_\beta(\omega)$.

Pour tout $(A_k)_{k\geq 0}$, nous notons $\mathfrak{I}((A_k)_{k\geq 0})$ la restriction de $\mathfrak{I}$ aux instructions définies sur $(A_k)_{k\geq 0}$.

1.3.2 Instructions classiques

Soit $k \geq 1$. Trois types d'instructions de niveau k sont généralement applicables aux it-piles :

- la **suppression** du sommet de pile de niveau k
- l'**empilement** d'un symbole au niveau k avec **copie** du sommet de pile de niveau $k-1$
- l'**échange** du symbole de sommet de pile de niveau k

Nous présentons ici ces trois d'instructions, ainsi qu'une quatrième, moins fréquemment utilisée, correspondant à l'instruction inverse de l'empilement.

Définition 1.3.1. *Soit $(A_\ell)_{\ell\geq 0}$ une suite d'alphabets de piles et $k \geq 1$. Les instructions* classiques *de niveau k sur $(A_k)_{k\geq 0}$ sont les instructions* pop_k, $(\mathrm{push}_{k,b})_{b\in A_k}$ *et* $(\mathrm{change}_{k,b})_{b\in A_k}$ *définies pour tout $\omega = a[\omega_1]\omega' \in k$-Pile, par :*

- $\mathrm{pop}_k(a[\omega_1]\omega') = \omega'$ *si* $a \neq \perp$, *sinon* $\mathrm{pop}_k(\omega)$ *est indéfini*
- $\mathrm{push}_{k,b}(a[\omega_1]\omega') = b[\omega_1]a[\omega_1]\omega'$
- $\mathrm{change}_{k,b}(a[\omega_1]\omega') = b[\omega_1]\omega'$, *si* $a \neq \perp$ *sinon* $\mathrm{change}_{k,b}(\omega)$ *est indéfini.*

Nous notons $\mathfrak{I}nstr$, la plus petite classe d'instructions contenant toutes les instructions pop_k, $\mathrm{push}_{k,b}$, $\mathrm{change}_{k,b}$ définies sur $(A_k)_{k\geq 0}$, pour $k \geq 1$, $(A_k)_{k\geq 0} \in \mathfrak{A}$

Remarquons que le niveau des instructions $\mathrm{push}_{i,b}$ et $\mathrm{change}_{i,b}$ est indiqué par celui de la lettre b. Ainsi, lorsqu'il n'y a pas d'ambiguïté sur le niveau de b, nous noterons de préférence ces instructions push_b et change_b.

Détaillons maintenant l'effet de l'application de ces instructions. L'application de l'instruction $\mathrm{pop}_i(\omega)$, pour $1 \leq i \leq k$, à une pile $\omega \in k$-Pile supprime le i-atome $\mathrm{p}_i(\omega)$, du sommet de ω. C'est à dire, en considérant la représentation planaire de ω, que le sous-arbre de profondeur i le plus à gauche est supprimé.

Exemple 1.3.2. *L'instruction* pop *:*
$\mathrm{pop}_3(\omega_{ex}) = \perp\ [a_2[a_1\ \perp]a_2[\perp]\ \perp_2]$
$\mathrm{pop}_2(\omega_{ex}) = a_3[b_2[a_1\ \perp]\ \perp_2]\ \perp\ [a_2[a_1\ \perp]a_2[\perp]\ \perp_2]$
$\mathrm{pop}_1(\omega_{ex}) = a_3[a_2[a_1\ \perp]b_2[a_1\ \perp]\ \perp_2]\ \perp\ [a_2[a_1\ \perp]a_2[\perp]\ \perp_2]$.
La figure 10 illustre cet exemple sur la représentation planaire de ω_{ex}.

L'application à une pile $\omega \in k$-Pile, de l'instruction $\mathrm{push}_{i,b}$, pour $1 \leq i \leq k$, ajoute le i-atome $b[\mathrm{p}_{i-1}(\omega)]$, au niveau i du sommet de ω. C'est à dire, en considérant la représentation planaire de ω, que le sous-arbre de profondeur i le plus à gauche est recopié en sommet de pile, et l'étiquette de sa racine est remplacée par b.

Exemple 1.3.3. *L'instruction push :*
$\mathrm{push}_{3,b_3}(\omega_{ex}) = b_3[a_2[a_1a_1\ \perp]b_2[a_1\ \perp]\ \perp_2]a_3[a_2[a_1a_1\ \perp]b_2[a_1\ \perp]\ \perp_2]\ \perp\ [a_2[a_1\ \perp]a_2[\perp]\ \perp_2]$
$\mathrm{push}_{2,b_2}(\omega_{ex}) = a_3[b_2[a_1a_1\ \perp]a_2[a_1a_1\ \perp]b_2[a_1\ \perp]\ \perp_2]\ \perp\ [a_2[a_1\ \perp]a_2[\perp]\ \perp_2]$
$\mathrm{push}_{1,b_1}(\omega_{ex}) = a_3[a_2[b_1a_1a_1\ \perp]b_2[a_1\ \perp]\ \perp_2]\ \perp\ [a_2[a_1\ \perp]a_2[\perp]\ \perp_2]$.
La figure 11 illustre cet exemple sur la représentation planaire de ω_{ex}.

Enfin, l'application de l'instruction $\mathrm{change}_{i,b}$, pour $1 \leq i \leq k$, à $\omega \in k$-Pile consiste à remplacee le symbole de tête de niveau i par b. C'est à dire, en considérant la représentation planaire de ω, que l'étiquette de hauteur i la plus gauche est remplacée par b.

Exemple 1.3.4. *L'instruction* change *:*
$\mathrm{change}_{3,b_3}(\omega_{ex}) = b_3[a_3[a_1a_1\ \perp]a_2[a_1\ \perp]\ \perp_2]\ \perp\ [a_2[a_1\ \perp]a_2[\perp]\ \perp_2]$
$\mathrm{change}_{2,b_2}(\omega_{ex}) = a_3[b_2[a_1a_1\ \perp]a_2[a_1\ \perp]\ \perp_2]\ \perp\ [a_2[a_1\ \perp]a_2[\perp]\ \perp_2]$
$\mathrm{change}_{1,b_1}(\omega_{ex}) = a_3[a_2[b_1a_1\ \perp]a_2[a_1\ \perp]\ \perp_2]\ \perp\ [a_2[a_1\ \perp]a_2[\perp]\ \perp_2]$.

Remarque 1.3.5.

1. *Remarquons que dès le niveau $i \geq 2$, les instructions de type* pop *et* push *ne sont plus symétriques. Ainsi, la plupart du temps, pour ω donné, il n'existe pas de lettre $a \in A_i$*

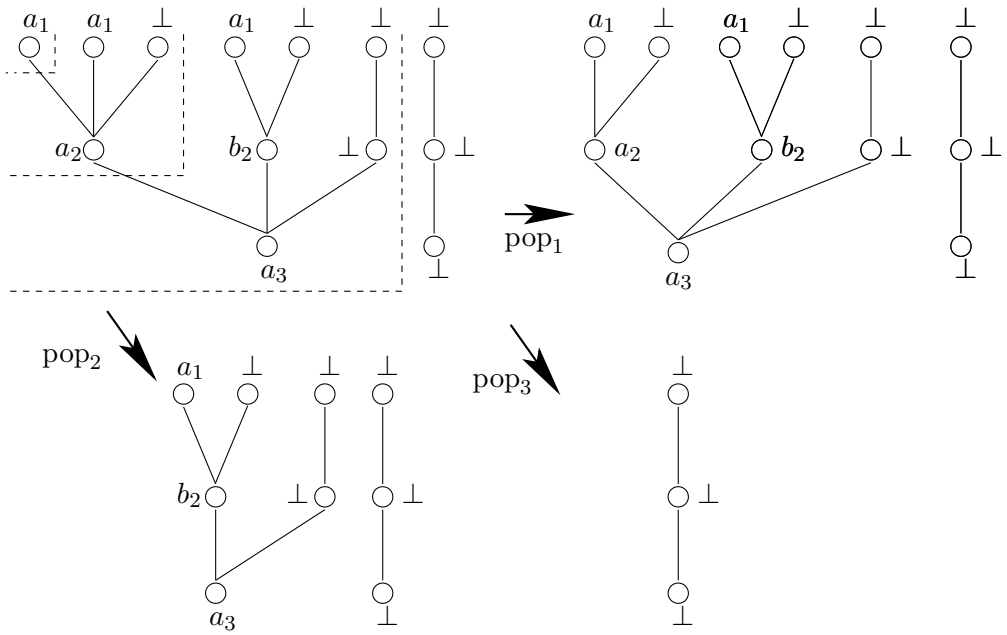


FIG. 10 – L'instruction pop

telle que $\text{push}_{i,a}(\text{pop}_i(\omega)) = \omega$. *Par contre, l'égalité* $\text{pop}_i(\text{push}_{i,a}(\omega)) = \omega$ *est toujours vraie.*

2. *L'instruction* $\text{change}_{1,b}$ *n'est pas indispensable puisqu'elle correspond à appliquer* pop_1 *suivit de* $\text{push}_{1,b}$, *par contre, au niveau* $i \geq 2$, *l'instruction* change_i *n'est pas directement simulable.*
3. *Étant donné* $\text{instr} \in \mathfrak{I}nstr_k$ *et* ω *une* k*-pile, pour savoir si* $\text{instr}(\omega)$ *est défini, il suffit de considérer la liste de symboles* $\text{top}(\omega)$.

Nous définissons une quatrième instruction qui n'est usuellement pas utilisée sur les *it*-piles et qui correspond à l'instruction symétrique de $\text{push}_{i,a}$:

Définition 1.3.6. *Soit* $(A_i)_{i\geq 0}$ *et* $k \geq 1$. *Pour tout* $b \in A_k$, *la* k*-instruction* $\overline{\text{push}}_{k,b}$ *est définie pour tout* $\omega \in k\text{-Pile}(A_1, \ldots, A_k)$ *par*

- $\overline{\text{push}}_{k,b}(\omega) = \text{pop}_i(\omega)$ *si il existe* $a \in A_k^{\perp}$, $\omega_1 \in (k-1)\text{-Pile}$ *et* $\omega' \in k\text{-Pile} \cup \{\varepsilon\}$ *tels que* $\omega = b[\omega_1]a[\omega_1]\omega'$
- $\overline{\text{push}}_{k,b}(\omega)$ *est indéfini sinon.*

Nous notons $\overline{\mathfrak{I}nstr}$ la classe des instructions $\text{push}_{k,b}$ et $\overline{\text{push}}_{k,b}$ pour $k \geq 1$, $b \in A_i$ et $(A_i)_{i\geq 0} \in \mathfrak{A}$.

Une définition équivalente de $\overline{\text{push}}_{i,b}$ est la suivante : $\forall \omega \in k\text{-Pile}$, $\overline{\text{push}}_{i,b}(\omega)$ est défini et égal à ω' ssi $\omega = \text{push}_{i,b}(\omega')$.

Remarque 1.3.7. *Lorsqu'elle est définie, l'instruction* $\overline{\text{push}}_{i,b}$ *donne donc de nombreuses informations sur la pile à laquelle elle est appliquée. En effet, si* $\overline{\text{push}}_{i,b}(\omega)$ *est défini, alors forcément,*

- $\text{top}_i(\omega) = b$,
- *si* $\omega \in k\text{-Pile}$, $k > i$, *alors* $\text{p}_i(\omega)$ *est de la forme* $b[\omega_1]c[\omega_1]\omega'$.

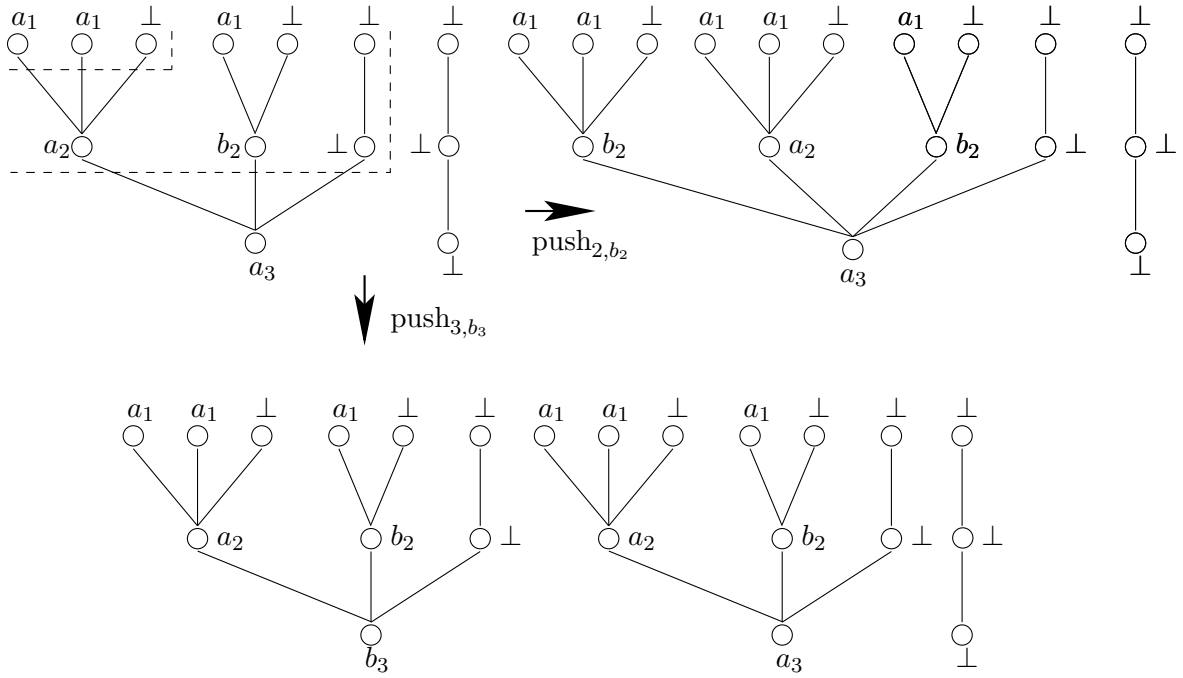


FIG. 11 – L'instruction push

– $\mathrm{push}_{i,b}(\mathrm{pop}_i(\omega)) = \mathrm{pop}_i(\mathrm{push}_{i,b}(\omega))$.

1.4 Représentation dans un groupe libre

Au niveau 1, l'ensemble des piles sur un alphabet A forme un monoïde isomorphe au monoïde libre A^* (pour le produit de concaténation définit §1.2.1). Plongeons A^* dans le groupe libre $(\mathrm{Irr}(A), \bullet, \varepsilon)$ (voir §0.1). Pour toute 1-pile ω, l'image par mot_1 de $\mathrm{push}_a(\omega)$ est exactement $\mathrm{mot}_1(\omega) \bullet a$. De même, lorsqu'elle est définie, l'image de $\overline{\mathrm{push}}_a(\omega)$ est $\mathrm{mot}_1(\omega) \bullet \bar{a}$.

L'extension de cette représentation au niveau k permet d'obtenir une représentation linéaire des k-piles. Nous codons chaque $\omega \in k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$, par un mot dans le groupe $(\mathrm{Irr}(A_{1,k}), \bullet, \varepsilon)$ décrivant la plus courte suite d'instructions $\mathrm{push}_{i,a}$ et $\overline{\mathrm{push}}_{i,a}$ calculant ω à partir de $\bot_k$. L'ensemble de ces codages est un ensemble de mots, noté $\mathcal{P}_k$ en bijection avec k-Pile.

L'idée d'une telle représentation est évoquée dans [Mas76] et utilisée dans [Car05], pour définir une notion de régularité sur les piles.

Soit $(A_\ell)_{\ell \geq 0}$ une suite d'alphabets de piles, pour tout $k \geq 1$, nous notons $A_{1,k}$ l'union des ensembles $A_1, \ldots, A_k$.

Chaque $\omega \in k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$ peut-être représenté par un mot sur $\widehat{A_{1,k}} = A_{1,k} \cup \overline{A_{1,k}}$ codant une suite d'instructions calculant ω à partir de $\bot_k$:

– chaque $a \in A_i$ correspond à $\mathrm{push}_{i,a}$

– chaque $\bar{a} \in \overline{A_i}$ correspond à $\overline{\mathrm{push}}_{i,a}$.

Par exemple, la 2-pile $\omega = a_2[a_1 b_1 \bot] a_2[a_1 \bot] \bot_2$ peut être représentée par le mot $u_1 =$

$a_2a_1a_2\bar{a}_1b_1a_1$, ou par $u_2 = a_2a_1b_2\bar{b_2}a_2\bar{a}_1b_1a_1$, ou encore par $u_2 = a_2a_1a_1b_2\bar{b_2}\bar{a_1}a_2\bar{a}_1b_1a_1$.

Il y a donc une plusieurs représentations de la même k-pile mais toutes ont la même réduction dans le groupe libre. Chaque k-pile sera donc codée par sa représentation réduite. Dans l'exemple précédent le codage est u_1 puisque $u_1 = \rho(u_1) = \rho(u_2) = \rho(u_3)$.

Remarquons de plus que tout mot dans $\widehat{A_{1,k}}^*$ ne définit pas forcément une séquence d'instructions *valide*. Par exemple, $a_1b_1a_2\bar{b_1}\bar{b_1}$ n'est pas valide puisque $a_1b_1a_2\bar{b_1}$ correspond à $\omega = a_2[a_1] \perp [b_1a_1]$ et $\overline{\text{push}}_{b_1,1}(\omega)$ est alors indéfini.

Nous définissons donc l'ensemble $\mathcal{M}_k(A_1, \ldots, A_k)$ (ou plus simplement $\mathcal{M}_k$ quand les A_i sont fixés) des mots de $\widehat{A_{1,k}}^*$ qui représentent une séquence valide de mouvements, ainsi que l'ensemble $\mathcal{P}_k(A_1, \ldots, A_k)$ (ou $\mathcal{P}_k$) des mots réduits de $\mathcal{M}_k$, qui code l'ensemble des k-piles : $\mathcal{M}_k$ est l'ensemble des mots dont la réduction de tout préfixe appartient à $\mathcal{P}_k$, et $\mathcal{P}_{k+1}$ est l'ensemble des mots dont la projection dans $\widehat{A_{1,k}}^*$ appartient à $\mathcal{M}_k$. Formellement : $\mathcal{P}_0 = \{\varepsilon\}$, et $\forall k \geq 0$,

$$\begin{aligned} \mathcal{M}_k(A_1, \ldots, A_k) &= \{u \in \widehat{A_{1,k}}^* \mid \forall v \preccurlyeq u,\ \rho(v) \in \mathcal{P}_k(A_1, \ldots, A_k)\} \\ \mathcal{P}_{k+1}(A_1, \ldots, A_{k+1}) &= \{u \in (\overline{A_{1,k}} \cup A_{1,k+1})^* \mid \rho(u) = u,\ \pi_{\widehat{A_{1,k}}}(u) \in \mathcal{M}_k(A_1, \ldots, A_k)\} \end{aligned}$$

Par exemple, $\mathcal{P}_1(A_1) = {A_1}^*$ et
$\mathcal{P}_2(A_1, A_2) = \{u_0v_1a_1\overline{v_1}u_1 \cdots u_{n-1}v_na_n\overline{v_n}u_{n+1} \mid n \geq 0,\ u_i, v_i \in A_1^*, a_i \in A_2\}$.

La projection d'un mot de $\mathcal{P}_{k+1}$ dans $\mathcal{P}_k$ consiste à supprimer toutes les occurrences des lettres dans A_{k+1}, puis à appliquer la réduction ρ. Par définition de $\mathcal{P}_{k+1}$ et $\mathcal{M}_k$, le mot ainsi obtenu appartient à $\mathcal{P}_k$.

Définition 1.4.1 (Projection). *Soit $k \geq 0$, $\mathrm{f}_k : \mathcal{P}_{k+1} \to \mathcal{P}_k$ est défini pour tout $u \in \mathcal{P}_{k+1}$ par $\mathrm{f}_k(u) = \rho(\pi_{\widehat{A_k}}(u))$.*

Nous étendons f_k en une application de $\mathcal{P}_{k+1}$ dans $\mathcal{P}_i$, pour $i \leq k+1$. L'application $\mathrm{f}_{k+1,i} : \mathcal{P}_{k+1} \to \mathcal{P}_i$ est égale à la composition $\mathrm{f}_k \circ \mathrm{f}_{k-1} \circ \cdots \circ \mathrm{f}_i$ si $i \leq k$ et $\mathrm{f}_{k+1,k+1}$ est la fonction identité.

Remarque 1.4.2. *Tout $u \in \mathcal{P}_{k+1}$ admet une factorisation unique de la forme :*

$u = u_0v_1a_1\overline{v_1}u_1 \cdots u_{n-1}v_na_n\overline{v_n}u_{n+1}$ avec $a_1, \ldots, a_n \in A_{k+1}$, $n \geq 0$, $u_1, v_i \in \widehat{A_{1,k}}^*$ et $\mathrm{f}_k(u) = u_0u_1 \cdots u_{n+1}$.

Nous aurons fréquemment besoin de connaître la dernière lettre de la projection d'un mot de $\mathcal{P}_k$. Définissons donc une fonction indiquant cette lettre.

Définition 1.4.3. *Pour $0 \leq i \leq k$, l'application $\mathrm{fin}_i : \mathcal{P}_k(A_1, \ldots, A_k) \to \widehat{A_{1,i}} \cup \{\perp\}$ est définie pour tout $u \in \mathcal{P}_k$ par*

- *$\mathrm{fin}_i(u) = a$ si il existe w tel que $\mathrm{f}_{k,i}(u) = wa$,*
- *$\mathrm{fin}_i(u) = \perp$ si $\mathrm{f}_{k,i}(u) = \varepsilon$.*

La propriété suivante se démontre par une induction évidente sur k :

Proposition 1.4.4. *Pour tout $k \geq 1$, $u \in \mathcal{P}_k$ et $a \in A_i$, $1 \leq i \leq k$,*

- *$u \bullet a \in \mathcal{P}_k$,*
- *$u \bullet \bar{a} \in \mathcal{P}_k$ ssi $\mathrm{fin}_i(u) = a$.*

Les ensembles k-$\mathsf{Pile}(A_1, \ldots, A_k)$ et $\mathcal{P}_k(A_1, \ldots, A_k)$ sont liés par une bijection que nous notons φ_k.

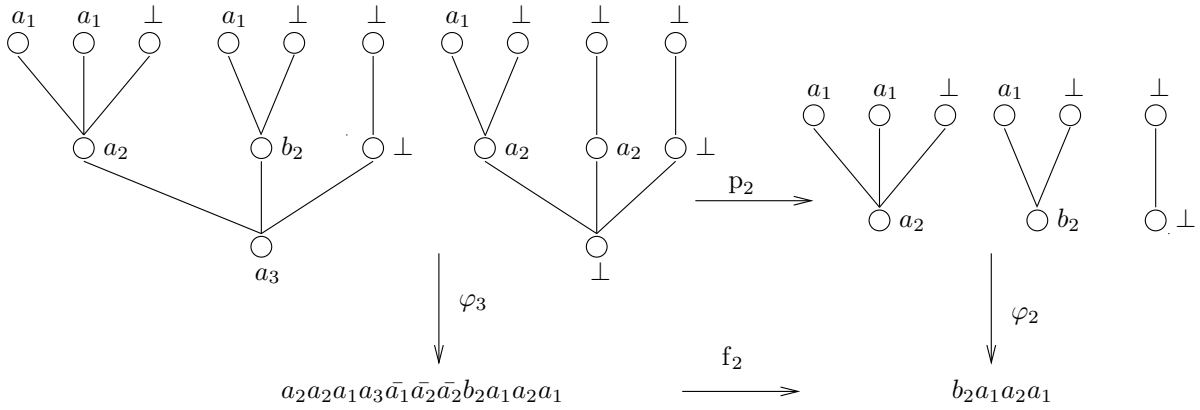


FIG. 12 – $\varphi_2(p_2(\omega_{ex})) = \mathrm{f}_2(\varphi_3(\omega_{ex}))$

Définition 1.4.5. *L'application* φ_k : k-Pile$(A_1, \ldots, A_k) \to \mathcal{P}_k(A_1, \ldots, A_k)$ *est définie par induction sur* $k \geq 0$ *:*

- $\varphi_0(\perp) = \varepsilon$,
- $\forall k \geq 1$, $\omega_1 \in k$-Pile, $\omega \in (k+1)$-Pile *et* $a \in A_{k+1}$,
  - $\varphi_{k+1}(\perp\ [\omega_1]) = \varphi_k(\omega_1)$,
  - $\varphi_{k+1}(a[\omega_1]\omega) = \varphi_{k+1}(\omega) \cdot a \cdot \overline{\mathrm{f}_k(\varphi_{k+1}(\omega))} \bullet \varphi_k(\omega_1)$.

Exemple 1.4.6. *Soit* ω_{ex} *la 3-pile donnée dans l'exemple 1.1.1, nous obtenons :*
$\varphi_3(\omega_{ex}) = a_2a_2a_1a_3\bar{a_1}\bar{a_2}\bar{a_2}b_2a_1a_2a_1$, *en effet,*
$\varphi_2(a_2[a_1a_1\ \perp]a_2[a_1\ \perp]\ \perp_2) = b_2a_1a_2a_1$ *et* $\varphi_2(a_2[a_1\ \perp]a_2[\perp]\ \perp_2) = a_2a_2a_1$, *alors*
$\varphi_3(\perp\ [a_2[a_1\ \perp]a_2[\perp]\ \perp_2]) = b_2a_2a_2a_1$ *et*
$\varphi_3(\omega_{ex}) = a_2a_2a_1a_3\overline{(a_2a_2a_1)} \bullet (b_2a_1a_2a_1) = a_2a_2a_1a_3\bar{a_1}\bar{a_2}\bar{a_2}b_2a_1a_2a_1$.

Proposition 1.4.7. *Pour tout* $\omega \in (k+1)$-Pile, $\varphi_k(p_k(\omega)) = \mathrm{f}_k(\varphi_{k+1}(\omega))$.

Cette propriété appliquée à un exemple est représentée par la figure 12.

Preuve : Par définition de φ_{k+1} et f_k,

$$\begin{aligned}
\mathrm{f}_k(\varphi_{k+1}(a[\omega_1]\omega)) &= \mathrm{f}_k(\varphi_{k+1}(\omega) \cdot a \cdot \overline{\mathrm{f}_k(\varphi_{k+1}(\omega))} \bullet \mathrm{f}_k(\varphi_k(\omega_1))) \\
&= \mathrm{f}_k(\varphi_{k+1}(\omega)) \bullet \overline{\mathrm{f}_k(\varphi_{k+1}(\omega))} \bullet \mathrm{f}_k(\varphi_k(\omega_1)) \\
&= \mathrm{f}_k(\varphi_k(\omega_1)).
\end{aligned}$$

□

Remarque 1.4.8. *En utilisant la proposition 1.4.7 et la définition de* φ_{k+1}, *il est clair que pour tout* $\omega = a_n[\omega_n] \cdots a_1[\omega_1]\ \perp\ [\omega_0] \in (k+1)$-Pile, $n \geq 0$,
$\varphi_{k+1}(\omega) = \varphi_k(\omega_0)a_1\overline{\varphi_k(\omega_0)} \bullet \varphi_k(\omega_1)a_2\overline{\varphi_k(\omega_1)} \bullet \varphi_k(\omega_2) \cdots a_n\overline{\varphi_k(\omega_{n-1})} \bullet \varphi_k(\omega_n)$.

La proposition 1.4.7 se généralise inductivement à tout $i \geq 0$, en utilisant la projection p_i : it-Pile $\to$ i-Pile définie dans le §1.1.

Proposition 1.4.9. *Pour tous* $1 \leq i \leq k$, *pour tout* $\omega \in k$-Pile, $\varphi_i(\mathrm{p}_{k,i}(\omega)) = \mathrm{f}_{k,i}(\varphi_k(\omega))$.

Vérifions maintenant que les ensembles k-Pile et $\mathcal{P}_k$ sont isomorphes.

Lemme 1.4.10. *Pour tout $k \geq 1$, φ_k est une bijection.*

Preuve :

1. Vérifions par induction sur $k \geq 0$ que φ_k est injective. Si $k = 0$ c'est évident. Soit $k \geq 0$, supposons que φ_k est une injection. Pour tous ω, $\omega' \in (k+1)$-Pile ayant la même image par φ_{k+1}, la remarque 1.4.8 implique clairement que ω et ω' se décomposent en $\omega = a_n[\omega_n] \cdots a_1[\omega_1] \perp [\omega_0]$ et $\omega' = a_n[\omega'_n] \cdots a_1[\omega'_1] \perp [\omega'_0]$.
   Vérifions que φ_{k+1} est injective par une seconde induction sur $n \geq 0$. Si $n = 0$, alors $\varphi_k(\omega_0) = \varphi_k(\omega'_0)$ et par hypothèse d'induction, $\omega_0 = \omega'_0$ et donc $\varphi_{k+1}(\omega) = \varphi_{k+1}(\omega')$. Si $n \geq 1$, posons $\omega'' = a_{n-1}[\omega_{n-1}] \cdots a_1[\omega_1] \perp [\omega_0]$ et $\omega''' = a_{n-1}[\omega'_{n-1}] \cdots a_1[\omega'_1] \perp [\omega'_0]$. Par définition de φ_{k+1} et puisque par hypothèse $\varphi_{k+1}(\omega) = \varphi_{k+1}(\omega')$:

$$\varphi_{k+1}(\omega'') \cdot a \cdot \overline{\mathrm{f}_k(\varphi_{k+1}(\omega''))} \bullet \mathrm{f}_k(\varphi_k(\omega_n)) = \varphi_{k+1}(\omega''') \cdot a \cdot \overline{\mathrm{f}_k(\varphi_{k+1}(\omega'''))} \bullet \mathrm{f}_k(\varphi_k(\omega'_n))$$

   c'est à dire :

$$\varphi_{k+1}(\omega'') = \varphi_{k+1}(\omega''') \text{ et } \overline{\mathrm{f}_k(\varphi_k(\omega''))} \bullet \mathrm{f}_k(\varphi_k(\omega_n)) = \overline{\mathrm{f}_k(\varphi_k(\omega'''))} \bullet \mathrm{f}_k(\varphi_k(\omega'_n)).$$

   Par hypothèse d'induction sur n, nous obtenons $\omega'' = \omega'''$ et puisque $(\mathrm{Irr}(A_{1,k}), \bullet, \varepsilon)$ est un groupe, l'inverse d'un mot réduit est unique et donc $\mathrm{f}_k(\varphi_k(\omega_n)) = \mathrm{f}_k(\varphi_k(\omega'_n))$. Donc $\omega = \omega'$, ce qui prouve que φ_k est injective pour tout $k \geq 0$.
2. φ_k est surjective et l'image inverse d'un mot de $\mathcal{P}_k$ est définie par :
   - ${\varphi_0}^{-1}(\varepsilon) = \perp$,
   - pour tout $u \in \mathcal{P}_{k+1}$, $k \geq 0$,
     - si $u \in \mathcal{P}_k$, alors ${\varphi_{k+1}}^{-1}(u) = \perp [{\varphi_k}^{-1}(u)]$,
     - sinon, il existe $u' \in \mathcal{P}_{k+1}$, $a \in A_{k+1}$, $u_1 \in \mathrm{Irr}(A_{1,k})$ tels que $u = u' \cdot a \cdot u_1$ et

$$ {\varphi_{k+1}}^{-1}(u) = a[{\varphi_k}^{-1}(\mathrm{f}_k(u))]{\varphi_{k+1}}^{-1}(u'). $$

   Une induction évidente permet de démontrer que ${\varphi_k}^{-1}$ est bien définie pour tout élément de $\mathcal{P}_k$ et correspond à l'application inverse de φ_k. Traitons le cas $u = u' \cdot a \cdot u_1 \in \mathcal{P}_{k+1}$, pour $a \in A_{k+1}$ et $u_1 \in \mathrm{Irr}(A_{1,k})$.

$$\begin{aligned} \varphi_{k+1}({\varphi_{k+1}}^{-1}(u)) &= \varphi_{k+1}(a[{\varphi_k}^{-1}(\mathrm{f}_k(u))]\varphi_{k+1}^{-1}(u')) \\ &= \varphi_{k+1}({\varphi_{k+1}}^{-1}(u')) \cdot a \cdot \overline{\mathrm{f}_k(\varphi_{k+1}({\varphi_{k+1}}^{-1}(u')))} \bullet \varphi_k(\varphi_k^{-1}(\mathrm{f}_k(u))) \\ &= u' \cdot a \cdot \overline{\mathrm{f}_k(u')} \bullet \mathrm{f}_k(u) = u' \cdot a \cdot \overline{\mathrm{f}_k(u')} \bullet \mathrm{f}_k(u') \bullet u_1 = u' \cdot a \cdot u_1. \end{aligned}$$

□

Remarque 1.4.11. *La remarque 1.4.8 implique directement que pour tout $\omega \in k$-Pile, $u \in \mathcal{P}_k$ avec $u = \varphi_k(\omega)$ pour tout $i \in [1, k]$,*

$$\mathrm{mot}_i(\omega) = \pi_{A_i}(\mathrm{f}_{k,i}(u))$$

En particulier, $\mathrm{mot}_k(\omega) = \pi_{A_k}(u)$ *et*
- $\mathrm{top}_i(\omega) = \perp$ *ssi* $\mathrm{fin}_i(u) = \varepsilon$,
- $\mathrm{top}_i(\omega) = a \in A_i$ *ssi il existe* $w \in \mathrm{Irr}(A_{1,i-1})$ *tel que* $\mathrm{fin}_i(u \bullet w) = a$.

Chapitre 2

Piles itérées comme mémoire

Nous nous intéressons maintenant aux machines utilisant les k-piles comme mémoire, et plus particulièrement aux *automates à k-piles*. Les automates à k-piles (k-AP) sont des systèmes de transitions sur les k-piles utilisant les instructions classiques : push, pop, change introduits au chapitre précédent. Nous étendons cette définition aux *automates à k-piles contrôlés* en autorisant les automates à effectuer des tests d'appartenance de la pile courante à des ensembles fixés appelés *contrôleurs*.

Nous avons déjà remarqué qu'il existait diverses variantes des piles itérées, il en est de même des automates. Afin d'englober toutes ces définitions, ainsi que des variantes comme les automates travaillant avec les instructions symétriques push et $\overline{\text{push}}$, nous choisissons de nous placer dans le cadre plus général des *machines* utilisant des k-piles comme mémoire.

Une machine à k-piles (k-MP) sur une classe $\mathfrak{I}$ d'instructions de k-piles, et contrôlée par un vecteur $\vec{C}$ d'ensembles de k-piles, est un système de transitions à mémoire dans k-Pile. Chaque transition utilise une instruction appartenant à $\mathfrak{I}$, et son application est conditionnée par l'état de la machine, la liste des symboles de tête de la mémoire et l'appartenance de la mémoire à chaque élément de $\vec{C}$. Cette définition contient donc les automates à k-piles (contrôlés), mais autorise également la définition de classes plus vastes utilisant d'autres instructions que push, pop et change.

Le choix de "prendre de la hauteur" par rapport aux automates à k-piles en définissant des machines plus générales présente deux principaux avantages. Tout d'abord, en offrant une "aire de comparaison" entre les différentes classes d'automates à it-piles, contrôlés ou non, ou entre les it-AP et d'autres classes utilisant des instructions de types différents. Nous introduirons la notion de "simulation déterministe", afin de comparer sous différent aspects les machines et classes de machines. Ceci nous permettra par exemple d'évaluer l'apport de l'utilisation de certains contrôleurs pour les automates à k-piles.

En second lieu, la flexibilité de la définition de machine permettra aux cours des nombreuses constructions à venir, d'utiliser diverses simplifications en utilisant la machine adéquate à chaque preuve, et de travailler ainsi par "quasi-isomorphisme" avec les k-AP.

Ainsi, bien que la plupart des définitions de ce chapitre seront posées dans le cadre général des machines à k-piles, l'objet principal de notre étude est les automates à k-piles (contrôlés) et la notion de machine est ici un outil de formalisation et de simplification.

2.1 Machines à piles itérées

Une machine à k-piles (k-MP) sur la famille d'instruction $\mathfrak{I}$, contrôlée par un vecteur $\vec{C}$ et avec entrées dans un alphabet fini Σ, est une machine abstraite dont les **configurations** sont des triplets (q, σ, ω). Le symbole q est l'état courant de la machine, $\sigma \in \Sigma^*$ est la partie du mot en entrée restant à calculer, et $\omega \in k\text{-}\mathsf{Pile}$ est la mémoire courante. La machine dispose d'un nombre fini de règles appelées **transitions**, et permettant de changer de configuration en lisant une lettre (possiblement ε) sur la bande d'entrée. L'application d'une transition à une configuration (q, σ, ω) envoie à une nouvelle configuration $(q', \sigma', \mathrm{instr}(\omega))$ pour $\mathrm{instr} \in \mathfrak{I}$. Cette application est conditionnée par l'appartenance de ω à chacun des éléments du vecteur de contrôle $\vec{C}$.

Définition

Soit $\mathfrak{I}$ une classe d'instruction de piles (voir §1.3.1), la classe $k\text{-}\mathsf{MP}(\mathfrak{I})$, des machines à k-piles à instructions dans $\mathfrak{I}$ est composé de toutes les structures

$$\mathcal{M} = (Q, \Sigma, (A_1, \ldots, A_k), \vec{C}, \Delta, q_0, \bot, F)$$

où

- Q est un ensemble fini représentant les *états* de la machine,
- Σ est un ensemble fini et désigne l'alphabet *d'entrée*,
- $A_1, \ldots, A_k$ sont les premiers éléments d'une suite d'alphabets $(A_\ell)_{\ell \geq 0} \in \mathfrak{A}$,
- $\bot$ est le symbole de fond de pile, et n'appartient donc à aucun des A_i,
- $\vec{C} = (C_1, \ldots, C_m)$, $m \geq 0$, $C_i \subseteq k\text{-}\mathsf{Pile}^{\bot}(A_1, \ldots, A_k)$ est le vecteur de *contrôleurs*.
- $q_0 \in Q$ est *l'état initial* de la machine,
- $F \subseteq Q$ est un ensemble *d'états finaux* et
- Δ est un sous-ensemble fini de *transitions* inclus dans

$$Q \times (\Sigma \cup \{\varepsilon\}) \times \mathrm{top}(k\text{-}\mathsf{Pile}^{\bot}(A_1, \ldots, A_k)) \times \{0,1\}^m \times \mathfrak{I}(A_1, \ldots, A_k) \times Q$$

Décrivons maintenant le mécanisme d'une telle machine.

- L'ensemble des **configurations** de $\mathcal{M}$ est $\mathrm{Conf}_{\mathcal{M}} = Q \times \Sigma^* \times k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$.
- La relation induite par $\mathcal{M}$ est notée $\vdash_{\mathcal{M}} \subseteq \mathrm{Conf}_{\mathcal{M}} \times \mathrm{Conf}_{\mathcal{M}}$, et contient l'ensemble des couples $(p, \alpha\sigma, \omega) \vdash_{\mathcal{M}} (q, \sigma, \mathrm{instr}(\omega))$ tels que $(p, \alpha, \mathrm{top}(\omega), \chi_{\vec{C}}(\omega), \mathrm{instr}, q) \in \Delta$ (où $\chi_{\vec{C}}(\omega)$ est le vecteur caractéristique de ω dans $\vec{C}$ défini dans le §0.2). La clôture réflexive et transitive de $\vdash_{\mathcal{M}}$ est notée $\vdash^*_{\mathcal{M}}$.
- Le **langage accepté par** $\mathcal{M}$ est

$$\mathrm{L}(\mathcal{M}) = \{\sigma \in \Sigma^* \mid \exists q \in F,\ \exists \omega \in k\text{-}\mathsf{Pile},\ (q_0, \sigma, \bot_k) \vdash^*_{\mathcal{M}} (q, \varepsilon, \omega)\}.$$

2.2 Automates à piles itérées

Les *automates à k-piles contrôlés* généralisent la classe des automates à k-piles très étudiée dans les années 70 et introduite par Maslov [Mas74] afin de généraliser les *grammaires indexées* [Aho68]. Les *langages indexés* définis par Aho sont caractérisés par une classe d'automates : les *nested stack automata* [Aho69]. Maslov itère cette construction en donnant une caractérisation des langages de niveau k utilisant des *multilevel stack automata* [Mas76]. Depuis, de nombreux

types d'automates ont été définis pour reconnaître cette même classe de langages. Citons les *iterated-pushdown automata* étudiés par Damm et Goerdt [DG86] et Engelfriet [Eng83, Eng91] ou plus récemment la classe des *higher-order automata* définie par Knapik, Niwinski et Urzyczyn [KNU02].

La définition utilisée ici est proche de celle donnée dans [DG86], et est étendue à l'utilisation de contrôleurs.

Automates finis

Les automates finis correspondent aux machines à 0-piles. L'ensemble 0-Pile est composé de l'unique élément $\bot$. Toute classe d'instructions restreinte aux 0-piles est donc équivalente à $\{\text{stay}\}$. Nous définissons l'ensemble des automates finis $\mathsf{AF} = 0\text{-}\mathsf{MP}(\{\text{stay}\})$. Pour une machine dans AF, toutes les informations relatives à la mémoire sont superflues puisque celle-ci est toujours $\bot_0$. Un automate fini est donc donné de la façon suivante :

$$\mathcal{A} = (Q, \Sigma, \Delta, q_0, F) \text{ avec } \Delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times Q$$

Les langages reconnus par ces automates sont bien connus : il s'agit des langages *réguliers* ou *reconnaissables*. Nous notons REC l'ensemble de ces langages.

Théorème 2.2.1 (Kleene [HU79]). *Pour tous alphabets Σ_1 et Σ_2,*

1. *tout langage dans $\mathsf{REC}(\Sigma_1)$ est reconnu par un automate dans $\mathsf{AF}(\Sigma_1)$,* déterministe *et* sans ε-transitions *(i.e., tel que la relation de transition Δ est une fonction $\Delta : Q \times \Sigma_1 \to Q$).*
2. *$\mathsf{REC}(\Sigma_1)$ est clos par union, intersection et complément.*
3. *$\pi_1(\mathsf{REC}(\Sigma_1 \times \Sigma_2))$ est inclus dans $\mathsf{REC}(\Sigma_1)$, i.e., $\forall L \in \mathsf{REC}(\Sigma_1 \times \Sigma_2)$, $\pi_1(L) \in \mathsf{REC}(\Sigma_1)$.*

2.2.1 Automates à piles itérées

La classe des automates à piles itérées non contrôlés correspond à celle des machines utilisant les instructions classiques pop, push et change de la classe $\mathfrak{I}nstr$ (voir §1.3.2) et contrôlés par le vecteur nul.

Définition 2.2.2 (Automate à k-piles). *La classe k-$\mathsf{AP}^{\vec{\emptyset}}$ des automates à k-piles (non contrôlés) est l'ensemble des machines*

$$\mathcal{A} = (Q, \Sigma, (A_1, \ldots, A_k), \vec{\emptyset}, \Delta, q_0, \bot, F) \in k\text{-}\mathsf{MP}(\mathfrak{I}nstr)$$

Nous notons $\mathsf{LANG}_k(\Sigma)$ l'ensemble des langages dans Σ^* reconnus par de tels automates, et appelons ces langages **langages de niveau** *k. Nous notons k-$\mathsf{AP}^{\vec{\emptyset}}(A_1, \ldots, A_k)$ la restriction de k-$\mathsf{AP}^{\vec{\emptyset}}$ aux automates travaillant sur les alphabets de piles $A_1, \ldots, A_k$.*

Remarque 2.2.3. *Une autre définition possible est de considérer des automates acceptant sur pile vide. Il est facile de remarquer que les classe de langages reconnus avec les deux modes d'acceptation sont les mêmes. (Voir par exemple [HU79] pour le niveau 1).*

La littérature est relativement pauvre en exemples de langages de niveau k. L'exemple classique de langage dans LANG_2 qui ne soit pas dans LANG_1 est $L_1 = \{\alpha^n\beta^n\gamma^n \mid n \geq 0\}$. Une méthode de calcul de ce langage par un 2-AP est illustrée par la figure 2.2.1 et correspond aux étapes suivantes. L'automate empile au niveau 1 de $\bot_2$, autant de symboles a_1 que le

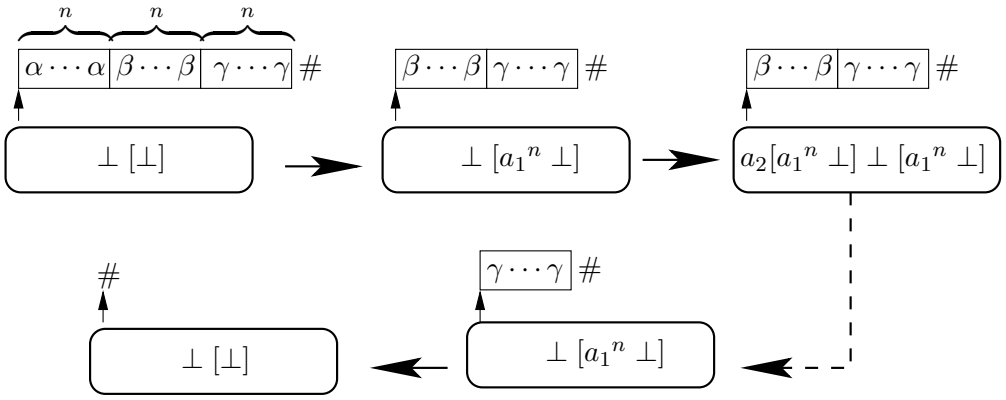


FIG. 13 – Un 2-AP reconnaissant le langage $\{\alpha^n\beta^n\gamma^n \mid n \geq 0\}$

préfixe de l'entrée contient d'occurrences du symbole α, puis il fait une copie de la 1-pile ainsi obtenue. La mémoire est alors de la forme $a_2[{a_1}^n \perp] \perp [{a_1}^n \perp]$. La première copie est alors utilisée pour compter le nombre de β en entrée, la seconde est ensuite utilisée à compter le nombre de γ terminant l'entrée.

Un autre exemple typique des possibilités d'un automate de niveau 2 est le langage $L_2 = \{\sigma \cdot \sigma \mid \sigma \in \Sigma^*\}$. Le principe de calcul de ce langage par un 2-AP est illustré sur un exemple par la figure 2.2.1.

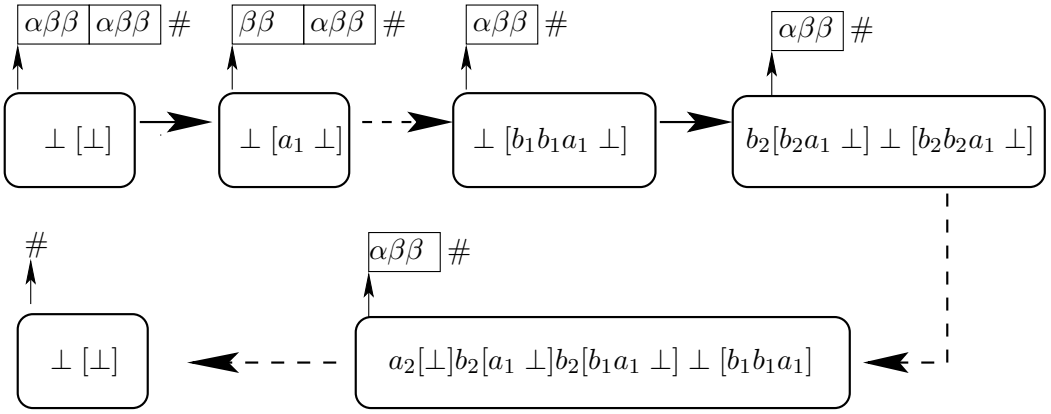


FIG. 14 – Un 2-AP reconnaissant le langage $\{\alpha^n\beta^n\gamma^n \mid n \geq 0\}$

Le premier σ est lu sur la bande d'entrée en empilant au niveau 1 de la pile, un représentant de chaque symbole lu (sur le schéma $\alpha \mapsto a_1$ et $\beta \mapsto b_1$). À la fin de la lecture de σ, le niveau 1 de la mémoire contient donc un représentant de l'image miroir de σ. Chaque symbole de niveau 1 est alors successivement dépilé et empilé au niveau 2 (avec la correspondance $a_1 \mapsto a_2$, $b_1 \mapsto b_2$), ceci jusqu'à obtention d'une pile ayant le sommet de niveau 1 vide. Notons ω la 2-pile ainsi obtenue, $\mathrm{mot}_2(\omega)$ est le représentant de l'image miroir de σ. Il suffit maintenant de vérifier que le reste de l'entrée est un second exemplaire de σ en dépilant successivement le niveau 2 jusqu'à obtenir une pile vide.

Nous donnons maintenant un exemple complet et moins habituel en définissant un automate reconnaissant les mots ayant la longueur d'un nombre de Fibonacci.

Exemple 2.2.4. *Le 2-AP suivant vérifie* $\mathrm{L}(\mathcal{A}) = \{\alpha^{f(n)} | \ n \geq 0\}$, *où* f *est la suite de Fibonacci définie par* $f(0) = f(1) = 1$ *et* $f(n+2) = f(n+1) + f(n)$, $\forall n \geq 0$.

$\mathcal{A} = (\{q_0, q, q_1, q_2, q_F\}, \{\alpha\}, (\{a_1\}, \{a_2, b_2\}), \Delta, q_0, \bot, \{q_F\})$ *et* Δ *est composé des transitions suivantes :*

$\Delta(q_0, \varepsilon, \bot\bot) = \{(\mathrm{push}_{a_1}, q_0), (\mathrm{push}_{a_2}, q)\}$,
$\Delta(q, \varepsilon, \bot\ a_1) = \{(\mathrm{push}_{a_1}, q_0), (\mathrm{push}_{2,a_2}, q)\}$,
$\Delta(q, \alpha, b_2\ \bot) = \Delta(q, \alpha, a_2\ \bot) = \{(\mathrm{pop}_2, q)\}$
$\Delta(q, \varepsilon, b_2a_1) = \{(\mathrm{pop}_1, q_1)\}$, $\Delta(q, \varepsilon, a_2a_1) = \{(\mathrm{pop}_1, q_2)\}$,
$\Delta(q_1, \varepsilon, b_2a_1) = \Delta(q_1, \varepsilon, b_2\ \bot) = \{(\mathrm{push}_{a_2}, q)\}$,
$\Delta(q_2, \varepsilon, a_2a_1) = \Delta(q_2, \varepsilon, a_2\ \bot) = \{(\mathrm{change}_{b_2}, q)\}$,
$\Delta(q, \varepsilon, \bot\ a_1) = \{(\mathrm{push}_{a_1}, q_F)\}$.

Voici une suite acceptante de configurations pour $\alpha^{f(3)} = a^3$ *:*

$(q_0, \alpha^3, \bot\ [\bot]) \vdash (q_0, \alpha^3, \bot\ [a_1\ \bot]) \vdash (q_0, \alpha^3, \bot\ [a_1a_1\ \bot]) \vdash (q_0, \alpha^3, \bot\ [a_1a_1a_1\ \bot]) \vdash$
$(q, \alpha^3, a_2[a_1a_1a_1\ \bot]\ \bot\ [a_1a_1a_1\ \bot]) \vdash (q_2, \alpha^3, a_2[a_1a_1\ \bot]\ \bot\ [a_1a_1a_1\ \bot]) \vdash$
$(q, \alpha^3, b_2[a_1a_1\ \bot]\ \bot\ [a_1a_1a_1\ \bot]) \vdash (q_1, \alpha^3, b_2[a_1\ \bot]\ \bot\ [a_1a_1a_1\ \bot]) \vdash$
$(q, \alpha^3, a_2[a_1\ \bot]b_2[a_1\ \bot]\ \bot\ [a_1a_1a_1\ \bot]) \vdash (q_2, \alpha^3, a_2[\bot]b_2[a_1\ \bot]\ \bot\ [a_1a_1a_1\ \bot]) \vdash$
$(q, \alpha^3, b_2[\bot]b_2[a_1\ \bot]\ \bot\ [a_1a_1\ \bot]) \vdash (q, \alpha^2, b_2[a_1\ \bot]\ \bot\ [a_1a_1a_1\ \bot]) \vdash$
$(q_1, \alpha^2, b_2[\bot]\ \bot\ [a_1a_1a_1\ \bot]) \vdash (q, \alpha^2, a_2[\bot]b_2[\bot]\ \bot\ [a_1a_1a_1\ \bot]) \vdash$
$(q, \alpha, b_2[\bot]\ \bot\ [a_1a_1a_1\ \bot]) \vdash (q, \varepsilon, \bot\ [a_1a_1a_1\ \bot]) \vdash (q_F, \varepsilon, \bot\ [a_1a_1a_1\ \bot])$

Théorème 2.2.5 ([Eng91]). *Pour tout* $k \geq 0$, *la classe* LANG_k *est close par homomorphisme, homomorphisme inverse et intersection avec un ensemble régulier et strictement incluse dans la classe* LANG_{k+1}.

2.2.2 Automates à piles itérées contrôlés

Nous travaillerons fréquemment sur une classe de machines plus vaste permettant d'effectuer des contrôles sur la pile. Nous appelons ces machines *automates contrôlés.*

Définition 2.2.6. *La classe* k-AP *des automates à* k*-piles contrôlés est l'ensemble des machines*

$$\mathcal{A} = (Q, \Sigma, (A_1, \ldots, A_k), \vec{C}, \Delta, q_0, \bot, F) \in k\text{-}\mathsf{MP}(\mathfrak{I}nstr)$$

Nous notons k-$\mathsf{AP}(A_1, \ldots, A_k)$ *la restriction de* k-AP *aux automates travaillant sur les alphabets de piles* $A_1, \ldots, A_k$, *et pour tout vecteur* $\vec{C}$ *de sous-ensembles de* k-$\mathsf{Pile}(A_1, \ldots, A_k)$, k-$\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k)$ *désigne la restriction de* k-$\mathsf{AP}(A_1, \ldots, A_k)$ *aux automates contrôlés par* $\vec{C}$.

Remarque 2.2.7. *La classe* k-$\mathsf{AP}^{\vec{\emptyset}}$ *des automates non contrôlés est donc une sous-classe de* k-AP.

Exemple 2.2.8. *Soit* $A_1 = \{a, b\}$, *et* $C = \{b^n a^n\ \bot \in 1\text{-}\mathsf{Pile}(A_1) \mid n \geq 1\}$. *Considérons l'automate* $\mathcal{A} \in 1\text{-}\mathsf{AP}^C(A_1)$ *donné par*

$$\mathcal{A} = (\{q_0, q_1, q_F\}, \{\alpha, \beta, \gamma\}, A_1, C, \Delta, q_0, \bot, \{q_f\})$$

où
$\Delta(q_0, \alpha, \bot, 0) = \Delta(q_0, \alpha, \bot, 1) = (\mathrm{push}_a, q_0)$,
$\Delta(q_0, \alpha, a, 0) = \Delta(q_0, \alpha, a, 1) = (\mathrm{push}_a, q_0)$,

$\Delta(q_0, \beta, a, 0) = \Delta(q_0, \beta, b, 0) = (\text{push}_b, q_0)$,
$\Delta(q_0, \varepsilon, \bot, 0) = \Delta(q_0, \varepsilon, \bot, 1) = (\text{stay}, q_F)$.

Cet automate reconnaît le langage $L(\mathcal{A}) = \{\alpha^n\beta^n\gamma^n,\ n \geq 1\} \in \mathsf{LANG}_2$.

2.3 D'autres machines

Nous introduisons ici des variantes de la définition d'automates à k-piles. Nous définissons tout d'abord des restrictions de k-AP, correspondant pour la plupart (à isomorphisme près) aux autres définitions "d'automate à piles itérées" existantes. Nous présentons ensuite les machines utilisant les instructions symétriques push et $\overline{\text{push}}$. La comparaison de ces différentes classes avec la classe k-AP sera étudiée dans la partie III.

2.3.1 Automates à piles itérées normalisés

Automates à piles N-normalisées

La première normalisation consiste à considérer des automates ne *générant* que des k-piles N-normalisées (correspondant aux k-piles "sans fonds de piles" (voir définition 1.2.1)). Cette sous-classe est équivalente (pour la version sans contrôleurs) à la notion d'automate définie dans [DG86] et utilisée dans [FS03].

Afin de ne générer que des piles N-normalisées, il suffit d'interdire les transitions amenant à appliquer $\text{push}_{i-1,a}$ à une pile ayant $\bot_i$ comme projection dans i-Pile.

Définition 2.3.1. *Notons k-NAP l'ensemble des automates à k-piles N-normalisés défini comme l'ensemble des automates $\mathcal{A} \in k\text{-AP}(A_1, \ldots, A_k)$ tels que*

si $(q, \alpha, a_k \cdots a_{i+1} \bot \cdots \bot, \text{instr}, p) \in \Delta$ *alors* $\text{instr} \notin \{\text{push}_{j,a}\}_{1 \leq j < i,\, a \in A_j}$.

Ainsi, la seule instruction applicable à la pile $\bot_k$ est une instruction push de niveau k.

Clairement, pour toute configuration (q, σ, ω) atteinte par un tel automate, ω est dans k-NPile.

Automates à piles "d'ordre supérieur"

La classe d'automates définie ici correspond à isomorphisme près (pour la version sans contrôleurs) à la notion "d'automate d'ordre supérieur" utilisée par exemple dans [Cac03], [CW03] ou [KNU02]. Ces automates travaillent sur des alphabets de piles réduits à un éléments pour tous les niveaux supérieurs à 1, utilisent des instructions appartenant à $\mathfrak{I}nstr$ et ne peuvent tester que le symbole de tête de niveau 1 de la pile (c'est-à-dire qu'il n'est pas possible de tester si les niveaux supérieurs à 1 sont vides).

Définition 2.3.2. *La classe k-N1AP des automates N1-normalisés correspond à la restriction de k-AP aux automates $\mathcal{A} = (Q, \Sigma, (A_1, \ldots, A_k), \vec{C}, \Delta, q_0, \bot, F)$ vérifiant :*

il existe une partition de Δ telle que pour tout élément $\bar{\delta}$ de la partition, il existe $p, q \in Q$, $a_1 \in A_1 \cup \{\bot\}$, $\alpha \in \Sigma \cup \{\varepsilon\}$, $\text{instr} \in \mathfrak{I}nstr(A_1, \ldots, A_k)$ tels que

$$\bar{\delta} = \{(p, \alpha, a_k \cdots a_1, \vec{o}, \text{instr}, q) \mid a_k \in A_k^{\bot}, \ldots, a_2 \in A_2^{\bot}\}$$

Ainsi, pour exprimer les transitions d'une telle machine, il sera plus commode de considérer le représentant $\delta = (p, \alpha, a_1, \vec{o}, \text{instr}, q)$ de la classe d'équivalence $\bar{\delta}$. Tout automate dans $k\text{-N1AP}^{\vec{\emptyset}}(A_1, \ldots, A_k)$ est donc équivalent à un automate dans $k\text{-N1AP}^{\vec{\emptyset}}(A_1, B_2, \ldots, B_k)$, pour

$B_2, \ldots, B_k$ réduits à un unique élément, puisque ces symboles n'influent pas sur l'application des transitions. Par contre, à moins de se restreindre à des contrôleurs particuliers, l'utilisation de contrôleurs fait ré-intervenir la valeur des lettres des niveaux supérieurs à 1.

Automate à instructions restreintes

Pour le niveau 1, l'instruction change n'est pas indispensable puisqu'elle est simulable par l'application de instruction pop suivie d'une instruction push.

C'est donc naturellement que nous nous intéressons à la classe des automates à k-piles n'utilisant pas ce type d'instruction.

La classe k-N2AP des automates N2-normalisés correspond à la restriction de k-AP aux automates dont les transitions n'utilisent jamais d'instruction $\text{change}_{i,a}$.

Remarque 2.3.3. *Clairement pour tout $k \geq 0$, les classes k-NAP, k-N1AP et k-N2AP sont incluses dans la classe k-AP. Par contre, pour $k \geq 2$, ces trois classes sont incomparables (pour la relation d'inclusion).*

2.3.2 Automates piles itérées "symétrique"

Rappelons que $\overline{\mathfrak{I}nstr}$ correspond à la classe des instructions de type push, et $\overline{\text{push}}$ (voir définition 1.3.6).

Définition 2.3.4 (Automate à instructions symétriques). *La classe $\overline{k\text{-AP}}$ des automates à k-piles symétriques est la classe k-MP($\overline{\mathfrak{I}nstr}$)*

2.4 Machines à piles itérées déterministes

La définition de l'ensemble des transitions d'une machine à k-piles autorise que plusieurs transitions soient appliquées à une même configuration. Nous définissons la sous-classe des k-MP **déterministes** comme l'ensemble des machines pour lesquelles indépendemment des contrôleurs choisis, ce cas n'arrive jamais. Ainsi, si un mot $\sigma \in \Sigma^*$ admet un calcul dans une machine déterministe, alors ce calcul est unique. Dans le cas des automates à 1-pile, la classe des automates déterministes est une des plus importante puisqu'il est possible de construire des analyseurs efficaces pour les langages correspondants, et de décider si deux automates reconnaissent le même langage [Sén02].

Définissons tout d'abord une condition pour que deux transitions soient applicables à une même configuration.

Définition 2.4.1. *Deux transitions $(p, \alpha, w, \vec{o}, \text{instr}, q)$ et $(p', \alpha', w', \vec{o'}, \text{instr}', q')$, avec $\vec{o}$, $\vec{o'} \in \{0,1\}^m$, $m \geq 0$ sont **compatibles** ssi $p = p'$, $w = w'$, $\vec{o} = \vec{o'}$ et*

$$[\alpha \neq \varepsilon \text{ et } \alpha' \neq \varepsilon \text{ et } \alpha = \alpha'] \text{ ou } [\alpha = \varepsilon] \text{ ou } [\alpha' = \varepsilon].$$

Définition 2.4.2 (Machine à k-piles déterministe). *Une machine à k-pile est dite **déterministe** si sa relation de transition ne contient pas de paires compatibles distinctes.*

2.5 Génération de graphes / d'ensembles de piles

Au delà du langage reconnu par une machine, il est intéressant de se pencher le fonctionnement plus "interne" de la machine, et en particulier sur la suite d'*états totaux* (i.e., couples (état, mémoire)) par lesquelles passe la machine lors d'un calcul. Une machine peut ainsi être vue comme un générateur d'ensembles de piles : elle génère l'ensemble des piles ω telles qu'il existe un calcul aboutissant dans un état total acceptant (p, ω) (i.e., p est un état final), ou comme le générateur du graphe dont les sommets sont les états totaux de la machine, et les arcs sont ceux induits par la relation de transition.

Définition 2.5.1. *Soit $\mathcal{M} = (Q, \Sigma, (A_1, \ldots, A_k), \vec{C}, \Delta, q_0, F, \bot) \in k\text{-}\mathsf{MP}(\mathcal{I})$ une machine quelconque. L'ensemble des états totaux de $\mathcal{M}$ est noté $\mathrm{EtatT}(\mathcal{M})$ et est composé de tous les couples $(p, \omega) \in Q \times k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$ pour lesquels il existe $\sigma \in \Sigma^*$ tel que*

$$(q_0, \sigma, \bot_k) \vdash^*_{\mathcal{M}} (p, \varepsilon, \omega).$$

2.5.1 Piles accessibles

Étant donnée $\mathcal{M} = (Q, \Sigma, (A_1, \ldots, A_k), \vec{C}, \Delta, q_0, \bot, F)$, une machine quelconque, le langage de k-piles généré par $\mathcal{M}$ est noté $\mathrm{ACC}(\mathcal{M})$. Il est composé de tous les $\omega \in k\text{-}\mathsf{Pile}$ pour lesquels il existe un état total (p, ω) acceptant (i.e., $p \in F$). Posons pour tout $Q' \subseteq Q$, $\mathrm{ACC}_{Q'}(\mathcal{M})$ est l'ensemble de tout les ω tels qu'il existe un état total (p, ω) tel que p appartient à Q' :

$$\mathrm{ACC}_{Q'}(\mathcal{M}) = \{\omega \in k\text{-}\mathsf{Pile}(A_1, \ldots, A_k) |\ \exists q \in Q',\ \exists \sigma \in \Sigma^*,\ (q_0, \sigma, \bot_k) \rightarrow^*_{\mathcal{M}} (q, \varepsilon, \omega)\}.$$

et

$$\mathrm{ACC}(\mathcal{M}) = \mathrm{ACC}_F(\mathcal{M})$$

Nous nous interesserons aux classes d'ensembles de piles générés par des automates à k-piles : nous notons $\mathsf{ACC}_k^{\vec{C}}(A_1, \ldots, A_k)$ l'union des $\mathrm{ACC}(\mathcal{A})$, pour $\mathcal{A}$ dans $k\text{-}\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k)$.

2.5.2 Graphe de calculs

Une autre façon d'interpréter une machine $\mathcal{M}$ est de considérer le graphe de relation induit par la machine. C'est un graphe orienté dont les arêtes sont étiquetées sur $\Sigma \cup \{\varepsilon\}$ et dont les sommets sont tous les couples (p, ω) accessibles depuis $(q_0, \bot)$.

Pour tous états totaux de $\mathcal{M}$, (p, ω) et (p', ω'), nous notons

- $(p, \omega) \overset{\sigma}{\rightarrow}_{\mathcal{M}} (p', \omega')$, $\sigma \in \Sigma^*$ si $(p, \sigma, \omega) \vdash_{\mathcal{M}} (p', \varepsilon, \omega')$,
- $(p, \omega) \overset{*}{\rightarrow}_{\mathcal{M}} (p', \omega')$ si il existe $\sigma \in \Sigma^*$ tel que $(p, \sigma, \omega) \vdash^*_{\mathcal{M}} (p', \varepsilon, \omega')$.

Définition 2.5.2. *Soit $\mathcal{M}$ une machine quelconque définie sur l'alphabet d'entrée Σ, le graphe de calcul de $\mathcal{M}$ est noté $\mathsf{Graphe}(\mathcal{M})$ et correspond au graphe*

$$(V, (E_\alpha)_{\alpha \in \sigma \cup \{\varepsilon\}})$$

dont les sommets sont $V = \mathrm{EtatT}(\mathcal{M})$, et pour tout $\alpha \in \Sigma \cup \{\varepsilon\}$, l'ensemble des arcs étiquetés par α est $E_\alpha = \overset{\alpha}{\rightarrow}_{\mathcal{M}}$.

Chapitre 3

Logique sur des structures à piles itérées

Nous regardons maintenant l'ensemble des k-piles comme le domaine de *structures relationnelles*. La logique que nous utilisons pour interpréter de telles structures est la logique du *Second Ordre Monadique* (SOM) qui est introduite dans la première section de ce chapitre. Trois types de structures en particulier retiendront notre attention :

- la structure associée à un arbre étiqueté
- la structure de domaine k-Pile induite par les graphes des instructions classiques
- la structure de domaine $\mathcal{P}_k$ (voir le chapitre précédent) induite par les graphes des relations "produit à droite" (dans le groupe libre) par une lettre de l'alphabet

Ces structures ne sont pas toutes des structures à k-piles mais contribueront à leur étude ; leur définition fait l'objet de la section 3.2. Nous terminons ce chapitre en énonçant les résultats importants existant sur la logique SOM de ces structures particulières, et des graphes de calculs des machines à k-piles.

3.1 Logique du Second Ordre Monadique (LSOM)

La logique du Second Ordre Monadique (SOM) est dotée d'un pouvoir d'expression puissant, puisqu'elle permet d'exprimer la clôture transitive d'une relation sur les éléments d'une structure relationnelle. De plus, grâce aux travaux de Büchi, puis de Rabin, il est connu que des structures importantes telles que les arbres complets admettent une théorie SOM. Ces caractéristiques font de cette logique un bon outil pour exprimer des propriétés portant sur des machines.

Considérons $Sig = \{r_1, \ldots, r_n\}$ une signature composé de symboles relationnels où chaque r_i est associé à une *arité* $(\rho_i, \tau_i) \in \mathbb{N}^2$. Une **structure** (relationnelle) $\mathcal{S}$ sur la signature Sig est composée d'un domaine $D_\mathcal{S}$ et de relations $r_1, \ldots, r_n$ sur $D_\mathcal{S}$.

Soit $Var = \{x, y, z, \ldots, X, Y, Z, \ldots\}$ un ensemble de variables où $x, y, z, \ldots$, désigne les variables du premier ordre et $X, Y, Z, \ldots$ celles du second ordre. L'ensemble SOM(Sig) des SOM-formules sur Sig est le plus petit ensemble tel que

- $x \in X$ et $Y \subset X$ sont des SOM-formules pour tous x, X, $Y \in Var$
- $r_i(x_1, \ldots, x_{\rho_i}, X_1, \ldots, X_{\tau_i})$ est une SOM-formule pour toutes variables $x_1, \ldots, x_{\rho_i}$ et $X_1, \ldots, X_{\tau_i} \in Var$

- si α et β sont des SOM-formules alors $\neg\alpha$, $\alpha \vee \beta$, $\exists x.\alpha$, $\exists X.\alpha$ sont des SOM-formules. Les variables x et X des formules $\exists x.\alpha$ et $\exists X.\alpha$ sont alors dites **liées**. Par opposition, toute variable non liée d'une formule sera dite **libre**.

Une SOM-formule est dite **close** si elle ne possède pas de variables libres. Étant donnée une formule Φ, nous noterons $\Phi(\bar{x}, \bar{X})$ (où $\bar{x} = (x_1, \ldots, x_\rho)$ et $\bar{X} = (X_1, \ldots, X_\tau)$ pour signifier que $x_1, \ldots, x_\rho$ sont les variables libres du premier ordre, et $X_1, \ldots, X_\tau$, celles du second ordre, de Φ.

Soit $\mathcal{S} = \langle D_\mathcal{S}, r_1, \ldots, r_n \rangle$ une structure définie sur la signature Sig, une **valuation** de Var dans $D_\mathcal{S}$ est une application $val : Var \to D_\mathcal{S} \cup 2^{D_\mathcal{S}}$ telle que $val(x) \in D_\mathcal{S}$, $\forall x, y, z, \ldots \in Var$ et $val(X) \subseteq D_\mathcal{S}$, $\forall X, Y, Z, \ldots \in Var$.

La satisfiabilité d'une SOM-formule Φ dans la structure $\mathcal{S}$ pour la valuation val est définie par induction sur la structure de la formule :

- $\mathcal{S}, val \models x \in X$ ssi $val(x) \in val(X)$, et $\mathcal{S}, val \models Y \subseteq X$ ssi $val(Y) \subseteq val(X)$,
- $\mathcal{S}, val \models r(x_1, \ldots, x_\rho, X_1, \ldots, X_\tau)$ ssi $r(val(x_1), \ldots, val(x_\rho), val(X_1), \ldots, val(X_\tau))$,
- si α et β sont des SOM-formules alors,
  $\mathcal{S}, val \models \neg\alpha$, ssi non$(\mathcal{S}, val \models \alpha)$,
  $\mathcal{S}, val \models \alpha \vee \beta$ ssi $\mathcal{S}, val \models \alpha$ ou $\mathcal{S}, val \models \beta$,
  $\mathcal{S}, val \models \exists x.\alpha$ ssi il existe $d \in D_s$ tel que $\mathcal{S}, [x \mapsto d, val] \models \alpha$,
  $\mathcal{S}, val \models \exists X.\alpha$ ssi il existe $S \subseteq D_s$ tel que $\mathcal{S}, [X \mapsto S, val] \models \alpha$.

Une SOM-formule ϕ sur Sig est dite **satisfiable dans** $\mathcal{S}$ si il existe une valuation val telle que $\mathcal{S}, val \models \phi$.

Nous noterons souvent pour simplifier $\mathcal{S} \models \phi(\bar{d}, \bar{S})$ à la place de $\mathcal{S}, [\bar{x} \mapsto \bar{d}, \bar{X} \mapsto \bar{S}] \models \phi(\bar{x}, \bar{X})$.

Définition 3.1.1. *La structure $\mathcal{S}$ admet une* SOM*-théorie* **décidable** *si pour toute* SOM*-formule close ϕ il existe une procédure effective permettant de décider si $\mathcal{S} \models \phi$.*

3.1.1 Ensembles SOM-définissables

Soit Sig une signature et $\mathcal{S}$ une structure de domaine $D_\mathcal{S}$ définie sur la signature Sig. Un ensemble $D \subseteq D_\mathcal{S}$ est dit **SOM-définissable** dans $\mathcal{S}$ si il existe une SOM-formule $\phi(X)$ telle que

$$\forall S \subseteq D_\mathcal{S}, \mathcal{S} \models \phi(S) \text{ ssi } S = D$$

Nous notons $\mathsf{DEF}(\mathcal{S})$ la classe des ensembles SOM-définissables dans $\mathcal{S}$.

Cette notion se généralise aux vecteurs d'ensembles. Le vecteur $\vec{S} = (S_1, \ldots, S_m)$, où les S_i sont des sous-ensemble de $D_\mathcal{S}$, est dit SOM-définissable si chacun des S_i appartient à $\mathsf{DEF}(\mathcal{S})$, ou de façon équivalente si il existe une formule $\phi(X_1, \ldots, X_n)$ telle que

$$\forall D_1 \ldots, D_m \subseteq D_\mathcal{S}, \mathcal{S} \models \phi(D_1, \ldots, D_m) \text{ ssi } S_1 = D_1, \cdots, S_m = D_m$$

Définition 3.1.2 (Propriété du modèle définissable). *Une structure $\mathcal{S}$ de domaine $D_\mathcal{S}$ définie sur une signature Sig vérifie la propriété du modèle définissable (MD) si pour toute formule $\phi(X_1, \ldots, X_n) \in$ SOM(Sig) satisfiable dans $\mathcal{S}$, il existe $\exists D_1, \ldots, D_n \subseteq D_\mathcal{S}$ tels que $\mathcal{S} \models \phi(D_1, \ldots, D_n)$ et chaque D_i est* SOM*-définissable dans $\mathcal{S}$.*

3.1.2 Interprétations sémantiques

Soit $Sig = \{r_1, \ldots, r_n\}$ et $Sig' = \{r'_1, \ldots, r'_m\}$ deux signatures relationnelles, $\mathcal{S}$ sur Sig et un ensemble de variables Var, de domaine $D_\mathcal{S}$ et $\mathcal{S}'$ sur Sig' et Var, de domaine $D_{\mathcal{S}'}$.

Définition 3.1.3. *Nous appelons* SOM-***interprétation*** *de la structure $\mathcal{S}$ dans la structure $\mathcal{S}'$ toute application injective $f : D_\mathcal{S} \to D_{\mathcal{S}'}$ telle que :*

1. *$f(D_\mathcal{S})$ est un ensemble* SOM-*définissable dans $\mathcal{S}'$*
2. *pour tout $i \in [1, n]$, il existe $\phi'_i(\overline{x}, \overline{X}) \in \mathrm{SOM}(Sig')$ (où $\overline{x} = x_1, \ldots, x_{\rho_i}$ et $\overline{X} = X_1, \ldots, X_{\tau_i}$) vérifiant pour toute valuation val de Var sur $D_\mathcal{S}$*

$$\mathcal{S}, val \models r_i(\overline{x}, \overline{X}) \Leftrightarrow \mathcal{S}', f \circ val \models \phi'_i(\overline{x}, \overline{X}).$$

Quand $D_\mathcal{S} \subseteq D_{\mathcal{S}'}$ et f est juste l'injection naturelle de $D_\mathcal{S}$ dans $D_{\mathcal{S}'}$, nous dirons que $\mathcal{S}$ est SOM-**définissable** dans $\mathcal{S}'$. Lorsque les structures sont SOM-interprétables l'une dans l'autre, nous dirons qu'elles sont SOM-**équivalentes**.

Le théorème suivant est démontré dans [Han77, §3.1 p. 613-615].

Théorème 3.1.4. *Si il existe une* SOM-*interprétation de $\mathcal{S}$ dans $\mathcal{S}'$, alors il existe une application calculable associant à toute formule close $\phi \in \mathrm{SOM}(Sig)$ une formule close $\hat{\phi} \in \mathrm{SOM}(Sig')$ telle que*

$$\mathcal{S} \models \phi \text{ ssi } \mathcal{S}' \models \hat{\phi}$$

En particulier, si $\mathcal{S}'$ a une SOM-*théorie décidable, alors $\mathcal{S}$ aussi.*

Rappelons que si E_1 et E_2 sont deux ensembles, alors pour tout couple $(e_1, e_2) \in E_1 \times E_2$, pour tout $i \in \{1, 2\}$, $\pi_i(e_1, e_2) = e_i$.

Définition 3.1.5. *Le* ***produit direct*** *$\mathcal{S} \times \mathcal{S}'$ est la structure définie sur la signature $Sig \cup Sig'$ de la façon suivante :*

1. *le domaine $D_{\mathcal{S}\times\mathcal{S}'}$ est $D_\mathcal{S} \times D_{\mathcal{S}'}$*
2. *pour tout $i \in [1, n]$, pour tous $d_1, \ldots, d_{\rho_i} \in D_{\mathcal{S}\times\mathcal{S}'}$ et $D_1, \ldots, D_{\tau_i} \subseteq D_{\mathcal{S}\times\mathcal{S}'}$,*

$$\mathcal{S} \times \mathcal{S}' \models r_i(d_1, \ldots, d_{\rho_i}, D_1, \ldots, D_{\tau_i}) \text{ ssi } \mathcal{S} \models r_i(\pi_1(d_1, \ldots, d_{\rho_i}), \pi_1(D_1, \ldots, D_{\tau_i}))$$

3. *pour tout $j \in [1, m]$, pour tous $d_1, \ldots, d_{\rho_i} \in D_{\mathcal{S}\times\mathcal{S}'}$ et $D_1, \ldots, D_{\tau_i} \subseteq D_{\mathcal{S}\times\mathcal{S}'}$,*

$$\mathcal{S} \times \mathcal{S}' \models r_j(d_1, \ldots, d_{\rho_i}, D_1, \ldots, D_{\tau_i}) \text{ ssi } \mathcal{S}' \models r_j(\pi_2(d_1, \ldots, d_{\rho_i}), \pi_2(D_1, \ldots, D_{\tau_i})).$$

Lemme 3.1.6. *Si $\mathcal{S}$ admet une* SOM-*théorie décidable et $\mathcal{S}$ est finie (i.e., son domaine est de cardinal fini), alors la structure $\mathcal{S} \times \mathcal{S}'$ a une* SOM-*théorie décidable.*

3.2 Structures relationnelles

Nous introduisons dans cette section les trois familles de structures relationnelles que nous rencontrerons le plus souvent : la structure d'arbre, la structure $\mathbf{Pile_k}$ de domaine k-Pile et correspondant au graphe des instructions instr_k, et la structure $\boldsymbol{\mathcal{P}_k}$ dont le domaine est $\mathcal{P}_k$ (défini §1.4) et les relations sont celles induites par le produit (par une lettre) dans le groupe libre.

Nous définissons ensuite différents modificateurs de structures. Ces définitions sont illustrées d'exemples sur nos trois structures de base..

3.2.1 Des structures remarquables

Définition 3.2.1. *Pour tous alphabets de piles $A_1, \ldots, A_k$, nous notons* $\mathbf{Pile_k}(A_1, \ldots, A_k)$ *la structure associée au graphe de relation induit par les instructions classiques :*

$$\mathbf{Pile_k} = \langle k\text{-Pile}, \bot_k, (\text{POP}_i, \text{PUSH}_a, \text{CHANGE}_a)_{1 \leq i \leq k, a \in A_i} \rangle$$

avec pour tout $1 \leq i \leq k$, pour tout $a \in A_i$:
- $\text{POP}_i := \{(\omega, \text{pop}_i(\omega)) \mid \omega \in k\text{-Pile}, \text{pop}_i(\omega)$ *défini* $\}$,
- $\text{PUSH}_a := \{(\omega, \text{push}_{i,a}(\omega)) \mid \omega \in k\text{-Pile}\}$,
- $\text{CHANGE}_a := \{(\omega, \text{change}_{i,a}(\omega)) \mid \omega \in k\text{-Pile}, \text{change}_a(\omega)$ *défini* $\}$.

Définition 3.2.2 (Structure d'arbre). *Pour tout alphabet A, la structure d'arbre complet non étiqueté est*

$$\mathbf{T}(A) = \langle A^*, \varepsilon, (\text{SUCC}_a)_{a \in A} \rangle$$

où $\forall a \in A$, $\text{SUCC}_a = \{(u, ua),\ u \in \{0,1\}^*\}$.

Remarquons que l'application mot_1 restreinte au piles de niveau 1 est une bijection et que les applications $\text{mot}_1 : \mathbf{Pile_1} \to \mathbf{T}(A)$ et $\text{mot}_1{}^{-1} : \mathbf{T}(A) \to \mathbf{Pile_1}$ sont des SOM-interprétations.

Définition 3.2.3 (Graphe des produits dans $\mathcal{P}_k$). *Pour tous alphabets de piles $A_1, \ldots, A_k$,*

$$\boldsymbol{\mathcal{P}}_\mathbf{k}(A_1, \ldots, A_k) = \langle \mathcal{P}_k, \varepsilon, (\bullet_a)_{a \in \widehat{A_k}} \rangle$$

où pour tout $a \in \widehat{A_k}$, la relation $\bullet_a$ est donnée par $\bullet_a := \{(u, u \bullet a) \mid u,\ u \bullet a \in \mathcal{P}_k\}$.

3.2.2 Modificateurs de structures

Nous utiliserons différents modificateurs pour construire de nouvelles structures à partir d'une structure donnée. Fixons $\mathcal{S} = \langle D_\mathcal{S}, r_1, \ldots, r_m \rangle$ sur une signature Sig.

Extension

Pour tous $O_1, \ldots, O_m \subseteq D_\mathcal{S}$, nous noterons $\mathcal{S}^{O_1,\ldots,O_m}$ la structure obtenue en enrichissant $\mathcal{S}$ des relations $(0,1)$-aires (nous dirons *unaires* pour simplifier) $O_1, \ldots, O_m$.

Cette nouvelle structure est donc définie sur la signature $Sig \cup \{O_1, \ldots, O_m\}$ et

$$\mathcal{S}^{O_1,\ldots,O_m} = \langle D_\mathcal{S}, r_1, \ldots, r_n, O_1, \ldots, O_m \rangle.$$

Exemple 3.2.4. *Pour tout $\vec{O} = (O_1, \ldots, O_m)$, avec $O_i \subseteq A^*$, la structure associée à l'arbre caractéristique* $\text{T}(A)^{\vec{O}}$ *(voir chapitre §0.4) est*

$$\mathbf{T}(A)^{\vec{O}} = \langle A^*, \varepsilon, (\text{SUCC}_a)_{a \in A}, O_1, \ldots, O_m \rangle.$$

Définissons la notion de formule à m places, qui nous permettra de passer facilement d'une formule sur une structure à une formule sur la même structure étendue.

Définition 3.2.5 (Formules à m places). *Une* SOM*-formule sur Sig à m places est une formule dans* $\text{SOM}(Sig \cup \{R_1, \ldots, R_m\})$ *où les R_i sont des symboles d'arité (0,1).*

Restriction

Pour tout $D \subseteq D_{\mathcal{S}}$, nous notons $\mathcal{S}_{|D}$ la structure sur Sig de domaine D, obtenue en restreignant les relations de $\mathcal{S}$ au nouveau domaine D.

Exemple 3.2.6. *Pour tout $\vec{O} = (O_1, \ldots, O_m)$, où $O_i \subseteq A^*$, et P sous-ensemble clos par préfixe de A^*, la structure associée à l'arbre caractéristique $\mathrm{T}(A)^{\vec{O}}{}_{|P}$ est*

$$\mathbf{T}(A)_{|P}{}^{\vec{O}} = \langle P, \varepsilon, (\mathrm{SUCC}_a)_{a\in A}, O_1 \cap P, \ldots, O_m \cap P\rangle$$

où pour tout $a \in A$, $\mathrm{SUCC}_a = \{(u, ua) |\ u, ua \in P\}$.

Image

Soit μ une application de domaine $D_{\mathcal{S}}$, la structure image de $\mathcal{S}$ par μ est définie sur Sig par

$$\mu(\mathcal{S}) = \langle \mu(D_{\mathcal{S}}), \mu(r_1), \ldots, \mu(r_m)\rangle$$

avec $\mu(r_i) = \{(\mu(d_1), \ldots, \mu(d_{\rho_i}), \mu(D_1), \ldots, \mu(D_{\tau_i}))|\ (d_1, \ldots, d_{\rho_i}, D_1, \ldots, D_{\tau_i}) \in r_i\}$.

Exemple 3.2.7. *Soit $|\ |_1 : \{0,1\}^* \to \mathbb{N}$ l'application associant à un mot u le nombre d'occurrences de 1 dans u,*

$$|\mathbf{T}(\mathbb{B})^{\vec{O}}|_1 = \langle \mathbb{N}, 0, |\mathrm{SUCC}_0|_1, |\mathrm{SUCC}_1|_1, |O_1|_1, \ldots, |O_m|_1\rangle \equiv \langle \mathbb{N}, 0, +1, |O_1|_1, \ldots, |O_m|_1\rangle$$

où $|\mathrm{SUCC}_0|_1 = \{(n, n),\ n \in \mathbb{N}\}$ et $|\mathrm{SUCC}_1|_1 = \{(n, n+1),\ n \in \mathbb{N}\}$.

Exemple 3.2.8. *Pour tout $1 \leq i \leq k$, et $C \subseteq k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$,*

$$\mathrm{p}_i(\mathbf{Pile_k}(A_1, \ldots, A_k)^C) \equiv \mathbf{Pile_i}(A_1, \ldots, A_i)^{\mathrm{p}_i(C)}.$$

Exemple 3.2.9. *Pour tout $1 \leq i \leq k$,*

$$\mathrm{f}_i(\mathbf{T}(\widehat{A_{1,k+1}})_{|\mathcal{P}_k}) \equiv \langle \mathcal{P}_i(A_1, \ldots, A_i),\ \varepsilon, (\bullet_a)_{a\in\widehat{A_j}}\rangle = \boldsymbol{\mathcal{P}_i}(A_1, \ldots, A_i).$$

3.3 Quelques résultats

3.3.1 Ensembles réguliers de piles 1-itérées

La logique SOM sur la structure $\mathbf{Pile_1}(A)$ (c'est-à-dire, par isomorphisme, sur l'arbre complet $\mathbf{T}(A)$) a déjà été largement étudiée. Nous résumons ici les principaux résultats.

Il est prouvé dans [Rab69] que la SOM-théorie de la structure $\mathbf{T}(A)$ est décidable. Par isomorphisme, nous avons donc :

Théorème 3.3.1 (Décidabilité). *Pour tout alphabet A, les structure $\mathbf{T}(A)$, $\mathbf{Pile_1}(A)$ admettent une* SOM*-théorie décidable.*

Il existe également des résultats sur la définissabilité. Toujours dans [Rab69], il est prouvé que les langages définissables dans $\mathbf{T}(A)$ sont exactement les langages réguliers.

Théorème 3.3.2 (Définissabilité). *Pour tout alphabet A,*

1. $\mathsf{DEF}(\mathbf{T}(A)) = \mathsf{REC}(A)$,
2. $\varphi_1(\mathsf{DEF}(\mathbf{Pile_1}(A))) = \mathsf{REC}(A)$.

Les ensembles définissables de $\mathbf{Pile_1}(A)$ ont également été comparés aux ensembles de piles générées par un automate à 1-piles. Il est prouvé dans [Gre67] qu'ils sont égaux.

Théorème 3.3.3 (Définissabilité). *Pour tout alphabet A,*

1. $\mathsf{DEF}(\mathbf{T}(A)) = \varphi_1(\mathsf{ACC}_1(A))$,
2. $\mathsf{DEF}(\mathbf{Pile_1}(A)) = \mathsf{ACC}_1(A)$.

3.3.2 Au delà du niveau 1

La décidabilité de la structure $\mathbf{Pile_k}$ est prouvée dans [FS03] par l'utilisation d'un théorème de transfert d'une structure vers sa "tree-structure". Étant donnée une structure $\mathcal{S} = \langle D, \mathrm{R}_1, \ldots, \mathrm{R}_n \rangle$, la "tree-structure" de S est

$$(\mathcal{S})^* = \langle D^*, \mathrm{SON}, \mathrm{CLONE}, \mathrm{R}_1^*, \ldots, \mathrm{R}_n^* \rangle$$

où $\mathrm{SON} = \{(w, wd) \mid d \in D\}$, $\mathrm{CLONE} = \{udd \mid d \in D\}$ et $\mathrm{R}_i^* = \{(wd_1, \ldots, wd_{\rho_i}) \mid (d_1, \ldots, d_{\rho_i}) \in \mathrm{R}_i\}$. Il est montré dans [MS92, Wal96] qu'une structure admet une SOM-théorie décidable ssi sa "tree-structure" a une SOM-théorie décidable. La structure $\mathbf{Pile_k}$ est SOM-interprétable dans la k-itération de cette transformation appliquée à $\mathbf{T}(A)$.

Proposition 3.3.4. *La* SOM*-théorie de* $\mathbf{Pile_k}$ *est décidable.*

Nous donnerons dans la partie II, une autre preuve de ce résultat, en utilisant une méthode fournissant également des informations sur la "structure" des ensembles définissables. Nous pouvons déjà constater que les piles accessibles d'un automate à k-piles forment un ensemble SOM-définissable dans la structure $\mathbf{Pile_k}(A_1, \ldots, A_k)$.

Définition 3.3.5. *Pour tout vecteur $\vec{C}$ de sous-ensembles de k-*$\mathsf{Pile}(A_1, \ldots, A_k)$*, nous notons* $\mathsf{DEF}_k^{\vec{C}}(A_1, \ldots, A_k)$ *la classe des ensembles* SOM*-définissables dans* ${\mathbf{Pile_k}}^{\vec{C}}(A_1, \ldots, A_k)$.

Proposition 3.3.6. *Pour tout vecteur $\vec{C}$ d'ensembles de k-piles sur les alphabets $A_1, \ldots, A_k$,*

$$\mathsf{ACC}_k{}^{\vec{C}}(A_1, \ldots, A_k)) \subseteq \mathsf{DEF}_k^{\vec{C}}(A_1, \ldots, A_k)$$

En particulier, si les composantes de $\vec{C}$ appartiennent à DEF_k*, alors*

$$\mathsf{ACC}_k{}^{\vec{C}}(A_1, \ldots, A_k) \subseteq \mathsf{DEF}_k(A_1, \ldots, A_k).$$

Preuve : Soit $\mathcal{A} = (Q, \Sigma, (A_1, \ldots, A_k), \vec{C}, \Delta, q_0, \bot, F)$ un automate à k-piles. Supposons que $Q = \{p_0, \ldots, p_n\}$, alors le vecteur $\vec{P} = (\mathrm{ACC}_{\{p_1\}}(\mathcal{A}), \ldots, \mathrm{ACC}_{\{p_n\}}(\mathcal{A}))$ est SOM-définissable dans ${\mathbf{Pile_k}}^{\vec{C}}$. En effet, posons

$$\mathrm{Acc}(X_1, \ldots, X_n) := \bigwedge_{i \in [1,n]} \forall x,$$

$$[x \in X_i \Leftrightarrow [[x = \bot \wedge p_i = q_0] \vee \bigvee_{\delta = (p_j, \alpha, a, \vec{o}, \mathrm{instr}, p_i)} \exists y \in X_j,\ \mathrm{TOP}_a(y) \wedge \chi_{\vec{C}}(y) = \vec{o} \wedge \mathrm{INSTR}(y, x)]]$$

($\mathrm{TOP}_a(\omega)$ correspond à la formule "$\exists \omega'$, $\omega = \mathrm{PUSH}_a(\omega')$".) Nous avons clairement ${\mathbf{Pile_k}}^{\vec{C}} \models \mathrm{Acc}(\vec{P})$. Toutefois, ce n'est peut-être par l'unique modèle de la formule. Nous allons montrer que c'est en fait le plus petit. Pour cela, il faut déjà prouver qu'il existe un plus petit modèle.

1. Pour tous modèles $\vec{S}$ et $\vec{S'}$ de Acc dans $\mathbf{Pile_k}^{\vec{C}}$, l'intersection de ces deux vecteurs est encore un modèle de la formule : $\mathbf{Pile_k}^{\vec{C}} \models \mathrm{Acc}(S_1 \cap S'_1, \ldots, S_n \cap S_n)$.
   Puisqu'il existe une solution, il existe une plus petite solution égale à l'intersection de tous les modèles dans $\mathbf{Pile_k}^{\vec{C}}$.
2. Pour tout vecteur $\vec{S}$ solution de Acc, pour tout $i \in [1, n]$, $P_i(\mathcal{A}) \subseteq S_i$. Ce résultat se vérifie par une induction évidente sur la longueur des plus courts calculs générant les éléments de P_i.

Puisqu'il existe une plus petite solution et que $\vec{P}$ est inclus dans toute solution, alors $\vec{P}$ est l'unique modèle dans $\mathbf{Pile_k}^{\vec{C}}$ de la formule suivante :

$\mathrm{Acc}^*(X_1, \ldots, X_n) :=$

$$\mathrm{Acc}(X_1, \ldots, X_n) \wedge \forall Y_1, \ldots, Y_n,\ \mathrm{Acc}(Y_1, \ldots, Y_n) \implies X_1 \subseteq Y_1 \wedge \cdots \wedge Y_n \subseteq X_n.$$

Finalement, l'ensemble $\mathrm{ACC}(\mathcal{A})$ est définissable puisqu'il correspond à l'union des P_i pour $p_i \in F$.

□

3.4 Logique sur les graphes de calculs

Nous étudions maintenant la décidabilité de la logique SOM sur les graphes de calculs des automates à k-piles (voir définition 2.5.2).

Théorème 3.4.1. *Si la structure* $\mathbf{Pile_k}(A_1, \ldots, A_k)^{\vec{C}}$ *admet une* SOM*-théorie décidable, alors pour tout automate* $\mathcal{A} \in k\text{-}\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k)$*, le graphe de calcul* $\mathsf{Graphe}(\mathcal{A})$ *admet une* SOM*-théorie décidable.*

Preuve : Considérons le produit direct de la structure $\mathbf{Pile_k}(A_1, \ldots, A_k)^{\vec{C}}$ par la structure finie $\mathcal{Q} = \langle Q, (E_q)_{q \in Q} \rangle$, où $E_{q,\mathcal{Q}} = \{q\}$. D'après le théorème 3.1.6, la structure $\mathcal{Q} \times \mathbf{Pile_k}(A_1, \ldots, A_k)^{\vec{C}}$ a une SOM-théorie décidable si $\mathbf{Pile_k}(A_1, \ldots, A_k)^{\vec{C}}$ a une SOM-théorie décidable.

Montrons que $\mathsf{Graphe}(\mathcal{M})$ est SOM-interprétable dans cette structure produit, ce qui démontrera donc le théorème.

– les arcs E_α correspondent à l'ensemble R_α des (x, y) tels que

$$\bigvee_{\delta=(p,\alpha,w,\vec{\sigma},\mathrm{instr},q)\in\Delta} x \in Q_p \wedge \mathrm{TOP}_w(x) \wedge \chi_{Q\times\vec{C}}(x) = \vec{\sigma} \wedge \mathrm{INSTR}(x, y) \wedge y \in Q_q,$$

– $(q_0, \bot_k)$ est définissable : $s_0(x) := Q_{q_0}(x) \wedge \bot_k(x)$,

– le domaine $\mathrm{EtatT}(\mathcal{A})$ de $\mathsf{Graphe}(\mathcal{A})$ correspond à la clôture transitive des relation correspondant à E_α et est donc définissable :

$$\mathrm{EtatT}(\mathcal{A}) = \{s \in Q \times k\text{-}\mathsf{Pile} \mid ((q_0, \bot), s) \in (\bigcup_{\alpha\in\Sigma\cup\{\varepsilon\}} R_\alpha)^*\}.$$

□

Corollaire 3.4.2. *Pour tout vecteur $\vec{C}$ d'éléments de $\mathsf{DEF}_k(A_1, \ldots, A_k)$, pour tout automate $\mathcal{A} \in k\text{-}\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k)$, le graphe de calcul $\mathsf{Graphe}(\mathcal{A})$ admet une* SOM*-théorie décidable.*

Preuve : La preuve est immédiate en utilisant le théorème 3.4.1 et la proposition 3.3.4.
□

Deuxième partie

Définissabilité dans une structure à piles itérées

Motivations

Initiée par les travaux de Büchi sur les mots, l'étude des liens entre les automates et la logique a mené à la construction de nombreuses structures ayant une SOM-théorie décidable. En particulier, Rabin prouve la décidabilité de la SOM-théorie de structures d'arbres infinis dans lesquelles de nombreuses théories et propriétés sont définissables.

Ces résultats ont également mené à la caractérisation suivante des langages réguliers. Les langages réguliers sur un alphabet A sont exactement :

- les langages reconnus par automate fini
- les ensembles SOM-définissables dans la structure $\mathbf{T}(A) = \langle A^*, \varepsilon, (\mathrm{SUCC}_a)_{a\in A}\rangle$

Les langages réguliers ont également été reliés aux automates à 1-piles, via la bijection naturelle existant entre les 1-piles et les mots (la bijection φ_1 définie au §1.4), induisant ainsi une notion de "régularité" pour les 1-piles. Les ensembles réguliers de 1-piles sur A sont exactement :

- les ensembles de 1-piles dont l'image par φ_1 est un langage régulier
- les ensembles définissables dans la structure $\mathbf{Pile_1} = \langle 1\text{-}\mathsf{Pile}(A), \bot, (\mathrm{push}_a)_{a\in A}, \mathrm{pop}_1\rangle$
- les ensembles de piles générés par un automate à 1-piles.

Au vu de ces résultats, il semble naturel d'introduire une notion de "régularité" sur les ensembles de k-piles, généralisant ces équivalences au niveau k. Nous avons montré dans la chapitre §1.4 que les k-piles étaient en bijection, par l'application φ_k, avec un langage de mots noté $\mathcal{P}_k$, c'est donc par le biais de ces mots que nous allons définir cette notion.

Les résultats de Rabin sur l'arbre complet $\mathbf{T}(A) = \boldsymbol{\mathcal{P}_1}(A)$ sont obtenus principalement grâce à la caractérisation par automates finis des langages réguliers. nous définissons donc une nouvelle famille *d'automates finis à oracles* notée AF_k permettant de reconnaître des langages dans $\mathcal{P}_k$. Nous obtenons ainsi une première hiérarchie, formée de classes de langages que nous appelons *k-réguliers*, pour $k \geq 0$, et admettant les deux caractérisations équivalentes suivantes. Les langages k-réguliers sont exactement :

- les langages reconnus par automate dans AF_k,
- les ensembles définissables dans la structure $\boldsymbol{\mathcal{P}_k} = \langle \mathcal{P}_k, \varepsilon, (\bullet_a)_{a\in\widehat{A_{1,k}}}\rangle$.

Ces résultats sont obtenus dans le cadre d'une étude détaillée des propriétés SOM de $\boldsymbol{\mathcal{P}_k}$, par le biais de l'étude plus générale des structures d'arbres étiquetés.

En reliant cette hiérarchie de classes de langages aux automates à k-piles, nous obtenons une seconde hiérarchie $(\mathsf{PREC}_k)_{k\geq 0}$, dont le niveau k correspond aux ensembles de k-piles "réguliers", et bénéficie de différentes caractérisations généralisant celles des ensembles de 1-piles réguliers. Les éléments de PREC_k sont exactement les ensembles de k-piles :

- dont l'image par φ_k est un langage k-régulier
- définissables en logique SOM dans la structure $\mathbf{Pile_k}$ correspondant au graphe des instructions de piles appliquées aux éléments de k-Pile
- générés par un automate à k-piles contrôlé par des ensembles $(k-1)$-réguliers.

Ces résultats sont obtenus grâce à l'étude plus générale de la logique sur les arbres étiquetés. Nous donnons une méthode de transfert, d'une structure de graphe, vers une structure d'arbre, des propriétés de décidabilité et de définissabilité.

Plan

Cette partie est découpée en trois chapitres.

Le premier chapitre introduit la notion d'automates finis à oracles d'arbres et de mots. Étant donné un vecteur $\vec{O} = (O_1, \ldots, O_m)$ de langages sur un alphabet fini A, un automate de mots à oracles $\vec{O}$ se comporte comme un automate fini, sauf que l'application des transitions est conditionnée par des tests d'appartenance des préfixes déjà calculés de l'entrée, à chacun des ensembles O_i. Les automates d'arbres sont similaires sauf que les tests portent sur le noeud courant de l'arbre en entrée. Nous étendons ensuite la correspondance établie par Rabin entre les forêts reconnaissables et les formules SOM sur la structure $\mathbf{T}(A)$ en prouvant que les forêts $\vec{O}$-reconnaissables (i.e., reconnues par automates à oracle $\vec{O}$) coïncident avec les forêts de modèles de formules SOM sur la structure étendue $\mathbf{T}(A)^{\vec{O}}$ (théorème 4.1.21). Nous obtenons ainsi une inter-réduction entre le problème du vide pour les forêts $\vec{O}$-reconnaissables et le problème de la satisfiabilité des SOM-formules closes sur $\mathbf{T}(A)^{\vec{O}}$.

Nous utilisons ensuite la théorie des jeux afin d'exprimer des propriétés portant sur les forêts $\vec{O}$-reconnaissables au moyen de SOM-formules sur la structure $\mu(\mathbf{T}(A)^{\vec{O}})$. Ceci permet d'obtenir les trois théorèmes annoncés précédemment.

Nous définissons dans le chapitre 5 la notion de langages de mots k-réguliers. Les théorèmes établis dans la section précédente permettent d'établir que la structure $\boldsymbol{\mathcal{P}}_{\mathbf{k}}$ admet une SOM-théorie décidable et satisfait la propriété MD (théorème 5.2.2), ainsi que de définir une classe AF_k d'automates reconnaissant exactement les ensembles SOM-définissables de $\boldsymbol{\mathcal{P}}_{\mathbf{k}}$ (les ensembles k-réguliers).

Nous établissons dans la section 3 des résultats similaires pour les k-piles. Nous prouvons pour cela que les structures $\boldsymbol{\mathcal{P}}_{\mathbf{k}}$ et $\mathbf{Pile}_{\mathbf{k}}$ sont SOM-équivalentes. Il s'ensuit que $\mathbf{Pile}_{\mathbf{k}}$ admet une SOM-théorie décidable et vérifie MD. Nous terminons cette section en prouvant le théorème 6.2.7 établissant l'égalité entre les ensembles SOM-définissables dans $\mathbf{Pile}_{\mathbf{k}}$, et les ensembles générés par des automates à k-piles contrôlés.

Chapitre 4

Définissabilité dans une structure arborescente

Ce chapitre est dédié à l'étude des propriétés de la logique SOM sur les structures d'arbre étiquetés

$$\mathbf{T}(A)_{|P}{}^{\vec{O}} = \langle P, (\mathrm{SUCC}_a)_{a\in A}, O_1, \ldots, O_m\rangle.$$

Nous nous intéressons en particulier aux trois propriétés suivantes :

- la décidabilité de la SOM-théorie,
- la "reconnaissabilité" par automates finis à oracles,
- l'existence d'un modèle définissable pour chaque formule satisfiable (c'est la propriété du Modèle Définissable (MD) introduite définition 3.1.2).

Un modèle dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ d'une formule satisfiable $\Phi(X_1, \ldots, X_n)$ est un vecteur $\vec{C} = (C_1, \ldots, C_n)$, avec $C_i \subseteq A^*$. Nous avons déjà remarqué qu'un tel vecteur pouvait être vu comme un arbre : l'arbre caractéristique de $\vec{C}$ dans A restreint à P, $\mathrm{T}(A)^{\vec{C}}_{|P} \in P\text{-}\mathsf{Arbre}(\{0,1\}^n)$.

L'ensemble des modèles dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ d'une formule à m variables libres peut donc être représentée par une forêt d'arbres de domaine P, étiquetés sur $\{0,1\}^n$. Rabin défini dans [Rab69], une classe d'automates **d'arbres** reconnaissant exactement les forêts de modèles de formules sur $\mathbf{T}(A)$. Les résultats de décidabilité étant obtenus plus facilement sur les automates que sur les formules, la correspondance établie par Rabin implique, via un travail sur les automates, la décidabilité de la structure $\mathbf{T}(A)$. De la même façon, en exhibant dans chaque forêt régulière (i.e., reconnue par un automate de Rabin) non vide, un arbre particulier dit *régulier*, Rabin prouve que chaque formule satisfiable dans $\mathbf{T}(A)$, admet un modèle $\vec{D}$ définissable en LSOM dans $\mathbf{T}(A)$ et tel que les composantes de $\vec{D}$ sont des langages de mots réguliers.

L'étude des forêts reconnaissables et des arbres réguliers est ainsi, dans le cas des arbres non étiquetés, à la base des résultats que nous cherchons à généraliser. Nous étendons donc la notion d'automates d'arbres en autorisant les automates à tester à l'aide d'un oracle, la valeur du noeud courant de l'arbre en entrée. Les forêts de domaine A^* reconnues par un automate à oracle $\vec{O}$ (un vecteur de langages dans A^*) sont appelées $\vec{O}$-reconnaissables.

Comme pour le cas des automates sans oracles, les forêts $\vec{O}$-reconnaissables sont exactement les ensembles de modèles de formules dans $\mathbf{T}(A)^{\vec{O}}$.

Remarquons que dans la plupart des cas, la SOM-théorie de $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ est indécidable (voir [Tho78] pour des exemples d'indécidabilité). Nous restreignons donc notre étude à des extensions $\vec{O}$ particulières. Nous utilisons les liens établis entre automates et formules pour transférer des SOM-propriétés vers un arbre $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ à partir de sa "structure image" par un morphisme. La notion de structure image a été introduite §3.2.2, pour $\mu : A^* \to M$ un morphisme dans un semi-groupe $(M, \star, e)$, l'image de $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ par μ est la structure $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ dont le domaine est $\mu(A^*)$ et les relations sont les images par μ des relations de $\mathbf{T}(A)_{|P}{}^{\vec{O}}$.

Nous utilisons la théorie des jeux pour prouver deux théorèmes de *transfert* de propriétés de $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ vers $\mathbf{T}(A)_{|P}{}^{\vec{O}}$, et procéder à une étude "structurelle" des ensembles définissables dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$.

Pour $\vec{O}$, P et μ "bien choisis", nous obtenons les résultats suivants :

- *Transfert de la décidabilité* (théorème 4.3.17) : si $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ a une théorie SOM décidable, alors $\mathbf{T}(A)^{\vec{O}}$ aussi.
- *Transfert de la propriété* MD (théorème 4.3.19) : si $\mu(\mathbf{T}(A)^{\vec{O}})$ satisfait la propriété du Modèle Définissable, alors $\mathbf{T}(A)^{\vec{O}}$ aussi.
- *Théorème de structure* (théorème 4.3.20) : si $\mu(\mathbf{T}(A)^{\vec{O}})$ satisfait MD, alors tout définissable de $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ est reconnu par un automate oracles de la forme $\mu^{-1}(D)$, pour D définissable en SOM dans $\mu(\mathbf{T}(A)^{\vec{O}})$.

Divers auteurs ont étudié les classes de prédicats unaires P pour lesquels $\langle \mathbb{N}, +1, P\rangle$ a une théorie SOM décidable (voir [ER66, CT02] et §13). Une application immédiate du premier théorème de transfert est que pour un tel prédicat, pour tout $B \subseteq A$, la structure $\mathbf{T}(A)^{l_B{}^{-1}(P)}$ a une SOM-théorie décidable (où $l_B^{-1}(P)$ est l'ensemble des mots de A^* dont le nombre d'occurrences de lettre dans B satisfait P).

Ce chapitre est découpé de la façon suivante : nous introduisons dans la première section les automates d'arbres à oracle et nous établissons la correspondance entre ces automates et les modèles de formules sur $\mathbf{T}(A)_{|P}{}^{\vec{O}}$. Dans le seconde section nous introduisons la notion d'arbre $\vec{O}$-*régulier* et étudions les liens avec la propriété du Modèle Définissable. La dernière section est dédiée aux preuves des théorèmes de transfert et de structure.

4.1 Forêt comme ensemble des modèles d'une formule

4.1.1 Automates de mots à oracles

Les automates finis à oracles étendent la classe des automates finis en autorisant des tests d'appartenance sur l'entrée de l'automate. Ces tests permettent d'aiguiller le choix des transitions à appliquer. Les oracles sont donnés par un vecteur $\vec{O} = (O_1, \ldots, O_m)$ de langages sur l'alphabet d'entrée A de l'automate. Les transitions sont similaires à celles d'un automate fini sauf qu'elles contiennent, en plus des informations habituelles, un vecteur booléen de taille m appelé **test** et indiquant les tests à effectuer sur l'entrée courante. Durant un calcul de $\mathcal{A}$,

la partie déjà calculée u du mot en entrée est gardée en mémoire et une transition avec test $\vec{o}$ peut être appliquée si $\vec{o}$ est le vecteur caractéristique de u dans $\vec{O}$ (i.e., si $\vec{o} = \chi_{\vec{O}}(u)$).

Définition 4.1.1 (Automate fini à oracles). *Un automate fini à oracles est la donnée d'un tuple $\mathcal{A} = (Q, A, \vec{O}, \Delta, q_0, F)$ où*

Q est l'ensemble (fini) des états
A est l'alphabet (fini) d'entrée
$\vec{O} = (O_1, \ldots, O_m)$, $O_i \subseteq A^*$, $m \geq 0$
$\Delta \subseteq Q \times A \times \{0,1\}^m \times Q$ est la relation de transition.
$q_0 \in Q$ est l'état initial,
$F \subseteq Q$ est l'ensemble des états finaux

- *Une configuration de $\mathcal{A}$ est un couple $(q, u_{\uparrow}v)$ où $uv \in A^*$ et $\uparrow$ est un symbole n'appartenant pas à A.*
- *La relation binaire sur les configurations induite par $\mathcal{A}$ est $\rightarrow_{\mathcal{A}}$ constituée de toutes les paires $(q, u_{\uparrow}av) \rightarrow_{\mathcal{A}} (p, ua_{\uparrow}v)$ telles que $(q, a, \chi_{\vec{O}}(u), p) \in \Delta$.*
- *Le langage reconnu par $\mathcal{A}$ est $\mathrm{L}(\mathcal{A}) = \{u \in A^* | \, (q_0, {}_{\uparrow}u) \rightarrow^*_{\mathcal{A}} (q_F, u_{\uparrow}), \, q_F \in F\}$.*
- *La famille d'automates sur A à oracle $\vec{O}$ est notée $\mathsf{AF}^{\vec{O}}(A)$ (ou $\mathsf{AF}^{\vec{O}}$ lorsqu'il n'y a pas de confusion possible).*
- *La classe des langages reconnus par automates dans $\mathsf{AF}^{\vec{O}}(A)$ est notée $\mathsf{REC}^{\vec{O}}(A)$. (Nous appellerons ces langages **$\vec{O}$-reconnaissables**).*
- *Nous étendons ces notations aux ensembles d'oracles : si $\mathcal{O}$ est un ensemble de vecteur de langages sur A, alors $\mathsf{AF}^{\mathcal{O}}(A)$ (resp. $\mathsf{REC}^{\mathcal{O}}(A)$) désigne l'union de tous les $\mathsf{AF}^{\vec{O}}(A)$ (resp. $\mathsf{REC}^{\vec{O}}(A)$) pour $\vec{O}$ dans $\mathcal{O}$.*

Remarque 4.1.2. *Les automates dans $\mathsf{AF}^{\vec{\emptyset}}$ sont exactement les automates finis. Nous noterons donc plutôt AF pour $\mathsf{AF}^{\vec{\emptyset}}$ et REC pour $\mathsf{REC}^{\vec{\emptyset}}$.*

Exemple 4.1.3. *Soit $\mathcal{A}_{ex} = (\{q_0, q_1, q_2, q_3, q_F\}, \{a, b, c\}, (O_1, O_2), \Delta, q_0, \{q_F\})$ où $O_1 = \{a^n b^n, \, n \geq 1\}$, $O_2 = \{a^* b^n c^{n-1}, \, n \geq 1\}$ et Δ est composé des transitions suivantes :*
$(q_0, a, (0,0), q_1)$, $(q_1, a, (0,0), q_1)$, $(q_1, b, (0,0), q_2)$, $(q_2, b, (0,0), q_2)$, $(q_2, b, (0,1), q_2)$,
$(q_2, c, (1,0), q_3)$, $(q_2, c, (1,1), q_F)$, $(q_3, c, (0,0), q_3)$, $(q_3, c, (0,1), q_F)$.
Le langage reconnu par cet automate est $\mathrm{L}(\mathcal{A}_{ex}) = \{a^n b^n c^n, \, n \geq 1\}$.

La relation de transition d'un automate dont le vecteur d'oracles est de taille m peut être représentée par un graphe fini dont les sommets sont les états de l'automate, et les arcs sont étiquetés sur $A \times \{0,1\}^m$. Ainsi, la relation de transition de l'automate $\mathcal{A}_{ex}$ défini dans l'exemple 4.1.3 correspond au le graphe représenté par la figure 15.

Associons à chaque automate $\mathcal{A} \in \mathsf{AF}^{\vec{O}}(A)$ un automate sans oracles $\widetilde{\mathcal{A}} \in \mathsf{AF}(A \times \{0,1\}^{|\vec{O}|})$, dit **source** de $\mathcal{A}$, obtenu en déplaçant le test de chaque transition dans la lettre à lire en entrée. La figure 16 représente l'automate source de l'automate $\mathcal{A}_{ex}$ donné figure 15.

Il apparaît clairement que $\mathrm{L}(\mathcal{A})$ correspond à la première projection de l'intersection de $\mathrm{L}(\widetilde{\mathcal{A}})$ avec un langage caractérisant $\vec{O}$.

Définition 4.1.4 (Langage caractéristique). *Pour tout vecteur $\vec{O}$ de langages dans A^*, de taille $|\vec{O}| = m$, le langage caractéristique de $\vec{O}$ est :*

$$L_\chi(A)^{\vec{O}} = \{(a_1, \vec{o}_1) \cdots (a_n, \vec{o}_n) \in (A \times \{0,1\}^m)^* | \; \forall i \in [1, n], \, \vec{o}_i = \chi_{\vec{O}}(a_1 \cdots a_{i-1})\}.$$

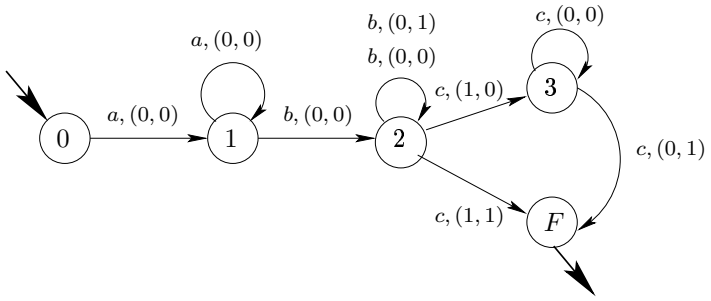


FIG. 15 – Les transitions d'un automate à oracles : $\mathcal{A}_{ex}$

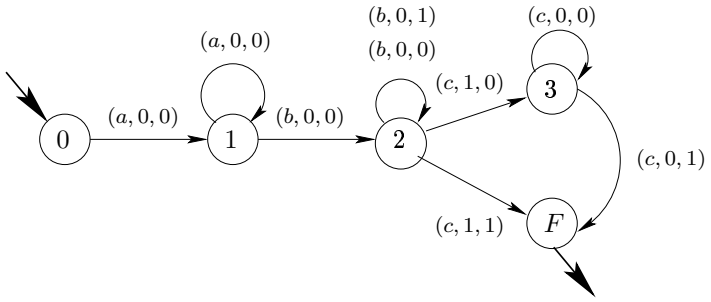


FIG. 16 – L'automate source de $\mathcal{A}_{ex}$

Lemme 4.1.5. *Pour tous $\mathcal{A} \in \mathsf{AF}^{\vec{O}}(A)$, $\vec{O} = (O_1, \ldots, O_m)$, $O_i \subseteq A^*$, et $\widetilde{\mathcal{A}} \in \mathsf{AF}(A \times \{0,1\}^m)$ tels que $\widetilde{\mathcal{A}}$ est* source *de $\mathcal{A}$,*

$$\mathrm{L}(\mathcal{A}) = \pi_1(\mathrm{L}(\widetilde{\mathcal{A}}) \cap L_\chi^{\vec{O}}(A)).$$

Puisque pour chaque $\vec{O}$ de taille m, tout automate dans $\mathsf{AF}(A \times \{0,1\}^m)$ est source d'un automate dans $\mathsf{AF}^{\vec{O}}(A)$, nous obtenons une caractérisation des langages reconnus par automates dans $\mathsf{AF}^{\vec{O}}$ à partir des langages réguliers :

Théorème 4.1.6. *Pour tout $\vec{O} \subseteq A^*$ de taille m :*

$$\mathsf{REC}^{\vec{O}}(A) = \pi_1(\mathsf{REC}(A \times \{0,1\}^m) \cap L_\chi^{\vec{O}}(A)).$$

Remarque 4.1.7.

1. *Si $\widetilde{\mathcal{A}}$ est l'automate source de $\mathcal{A}$, alors $\mathcal{A}$ est déterministe ssi $\widetilde{\mathcal{A}}$ déterministe.*
2. *Pour tout $\vec{O}$, le langage caractéristique $L_\chi^{\vec{O}}(A)$ est reconnu par un automate déterministe à oracle $(O_1 \times \{0,1\}^m, \ldots, O_m \times \{0,1\}^m)$.*

La caractérisation précédente permet de généraliser le théorème de Kleene (théorème 2.2.1) aux automates à oracles.

Théorème 4.1.8. *Pour tout alphabet A, pour tout oracle $\vec{O}$ sur A^* :*

1. *tout langage dans $\mathsf{REC}^{\vec{O}}(A)$ est reconnu par un automate* **déterministe** *dans $\mathsf{AF}^{\vec{O}}(A)$.*
2. *$\mathsf{REC}^{\vec{O}}(A)$ est clos par union, intersection et complément.*
3. *$\pi_1(\mathsf{REC}^{\vec{O}}(A \times B))$ appartient à $\mathsf{REC}^{\pi_1(\vec{O})}(A)$.*

Preuve :

1. Le premier point est évident par le lemme 4.1.5, la remarque 4.1.7(1) et le théorème 2.2.1.
2. Soient L_1 et $L_2 \in \mathsf{REC}^{\vec{O}}(A)$. D'après le lemme précédent, il existe deux langages réguliers $\widetilde{L_1}$ et $\widetilde{L_2}$ sur $A \times \{0,1\}^{|\vec{O}|}$ tels que

$$L_1 = \pi_1(\widetilde{L_1} \cap L_\chi^{\vec{O}}(A)) \text{ et } L_2 = \pi_1(\widetilde{L_2} \cap L_\chi^{\vec{O}}(A))$$

   Posons $L = \widetilde{L_1} \cap \widetilde{L_2}$, et $L' = \widetilde{L_1} \cup \widetilde{L_2}$, et $L'' = A^* - \widetilde{L_1}$. Par le théorème de Kleene (théorème 2.2.1), ces trois langages sont réguliers et nous avons clairement :

$$L_1 \cap L_2 = \pi_1(L \cap L_\chi^{\vec{O}}),\ L_1 \cup L_2 = \pi_1(L' \cap L_\chi^{\vec{O}}) \text{ et } A^* - L_1 = \pi_1(L'' \cap L_\chi^{\vec{O}}).$$

   En utilisant la caractérisation prouvée dans le théorème 4.1.6, ces trois langages appartiennent à $\mathsf{REC}^{\vec{O}}(A)$.
3. Soient A et B deux alphabets et $L \in \mathsf{REC}^{\vec{O}}(A \times B)$, pour $|\vec{O}| = m$. Il existe $\widetilde{L}$ dans $(A \times B \times \{0,1\}^m)$ tel que $L = \pi_{1\ldots2}(\widetilde{L} \cap L_\chi^{\vec{O}}(A \times B))$. Alors,

$$\pi_2(L) = \pi_1(\pi_{2,\ldots,m+2}(\widetilde{L} \cap L_\chi^{\vec{O}}(A))) = \pi_1(\pi_{2,\ldots,m+2}(\widetilde{L}) \cap L_\chi^{\pi_2(\vec{O})}(A)).$$

   Puisque REC est stable par projection, $\pi_{2,\ldots,n+2}(\widetilde{L})$ est régulier. En utilisant encore le théorème 4.1.6, il est clair que $\pi_2(L)$ appartient à $\mathsf{REC}^{\pi_2(\vec{O})}(B)$.

□

Remarque 4.1.9.

1. *Les langages reconnus par automate à k-piles étant stables par projection et intersection avec un rationnel (théorème 2.2.5), si $\mathcal{O}_k = \{\vec{O} |\ L_\chi^{\vec{O}} \in \mathsf{LANG}_k\}$, alors $\mathsf{REC}^{\mathcal{O}_k}$ est une partie de LANG_k stable par intersection et complémentation.*
2. *La définition choisie pour les automates à oracles n'autorise pas les ε-transitions, (i.e., qui lisent le mot vide sur la bande d'entrée). Tout automate dans $\mathsf{AF}^{\vec{O}}$ avec ε-transition n'est pas équivalent à un automate dans $\mathsf{AF}^{\vec{O}}$ sans ε-transitions, mais à un automate n'ayant que des ε-transitions vers un état final* "mort" *(i.e., duquel ne sort aucune transition), afin de faire un dernier test avant d'accepter un mot.*

4.1.2 Automates d'arbres à oracles

Nous définissons dans cette partie une extension des automates d'arbres similaire à celle introduite sur les automates de mots. Comme précédemment, pour un oracle $\vec{O}$ donné, l'application d'une transition à un sommet u d'un arbre est conditionnée par la valeur du vecteur caractéristique de u dans $\vec{O}$.

Définition 4.1.10 (Automate d'arbre à oracle). *Un automate d'arbre à oracle est une structure $\mathcal{A} = (Q, A, \Sigma, \vec{O}, \Delta, q_0, c)$ où Q, Σ et $A = \{a_1, \ldots, a_n\}$ sont des ensembles finis, $\vec{O}$ est un vecteur de m sous-ensembles de A^* (avec $m \geq 0$), $c : Q \to \{0, \ldots, max\}$, $max \geq 0$ et $\Delta \subseteq Q \times \Sigma \times \{0,1\}^m \times Q^n$.*

Étant donné $t \in A^*$-Arbre(Σ), un run *de $\mathcal{A}$ sur t est un arbre $r \in A^*$-Arbre(Q) vérifiant :*

- $r(\varepsilon) = q_0$
- $\forall u \in A^*$, $(r(u), t(u), \chi_{\vec{O}}(u), r(ua_1), \ldots, r(ua_n)) \in \Delta$.

Un run r est acceptant si pour tout chemin infini $\pi = q_1 \cdots q_n \cdots$ dans r, le plus petit $i \in [0, n_c]$ apparaissant infiniment souvent dans $c(q_1), \ldots, c(q_n), \ldots$ est pair.

La forêt dans A^*-Arbre(Σ) reconnue par $\mathcal{A}$ est notée $\mathrm{F}(\mathcal{A})$ et désigne l'ensemble des arbres pour lesquels il existe un run acceptant.

La classe des automates d'arbres dans A^*-Arbre(Σ), à oracle $\vec{O}$ est notée $\mathsf{AFA}^{\vec{O}}(A, \Sigma)$. L'ensemble des forêts reconnues par automates dans $\mathsf{AFA}^{\vec{O}}(A, \Sigma)$ est notée $\mathsf{TREC}^{\vec{O}}(A, \Sigma)$. Les forêt *reconnaissables* ou *régulières* correspondent à la classe $\mathsf{AFA}^{\vec{\emptyset}}$.

Comme pour les langages de mots, les forêts $\vec{O}$-reconnaissables sont obtenues par projection de l'intersection d'une forêt reconnaissable (i.e., reconnue par automate sans oracle) avec une forêt caractérisant le vecteur $\vec{O}$.

Définition 4.1.11 (Forêt caractéristique). *Étant donné un vecteur $\vec{O}$ de m langages sur A, et un ensemble fini Σ, la forêt caractéristique de $\vec{O}$ sur Σ est notée $F_\chi(A)^{\vec{O}}(\Sigma)$ et correspond à l'ensemble de tous les $t \in A^*$-Arbre$(\Sigma \times \{0,1\}^m)$ tels que $t \circ \pi_{1,m} = \mathrm{T}(A)^{\vec{O}}$.*

En d'autres mots, $F_\chi(A)^{\vec{O}}(\Sigma) = \{\mathrm{T}(A)^{\vec{O}}\}^\frown(A^*\text{-Arbre}(\Sigma))$, où $\mathrm{T}(A)^{\vec{O}}$ est l'arbre caractéristique de $\vec{O}$.

Le même type de construction que pour les mots (théorème 4.1.6) permet d'obtenir une caractérisation simple des forêts reconnaissables par automates dans $\mathsf{TREC}^{\vec{O}}(A, \Sigma)$. Associons à chaque automate $\mathcal{A} = (Q, A, \Sigma, \vec{O}, \Delta, q_0, c) \in \mathsf{AFA}^{\vec{O}}$, l'automate sans oracles $\widetilde{\mathcal{A}} = (Q, A, \Sigma \times \{0,1\}^m, \widetilde{\Delta}, q_0, c) \in \mathsf{AFA}$ où

$$\widetilde{\Delta} = \{(q, (\alpha, \vec{o}), p_0, p_1) \mid (q, \alpha, \vec{o}, p_0, p_1) \in \Delta\}.$$

Nous appelons $\widetilde{\mathcal{A}}$ l'automate **source** de $\mathcal{A}$. Il est facile de vérifier que

$$\mathrm{F}(\mathcal{A}) = \pi_1(\mathrm{F}(\widetilde{\mathcal{A}}) \cap F_\chi(A)^{\vec{O}}(\Sigma)).$$

Lemme 4.1.12. *Pour tout $\vec{O} = (O_1, \ldots, O_m)$, $O_i \subseteq A^*$,*

$$\mathsf{TREC}^{\vec{O}}(A, \Sigma) = \pi_1(\mathsf{TREC}(A, \Sigma \times \{0,1\}^m) \cap F_\chi(A)^{\vec{O}}(\Sigma)).$$

Remarque 4.1.13. *Pour tout $\vec{O} = (O_1, \ldots, O_m)$, $O_i \subseteq A^*$, les forêts $\{T(A)^{\vec{O}}\}$ et $F_\chi(A)^{\vec{O}}(\Sigma)$ sont $\vec{O}$-reconnaissables.*

Les forêts reconnaissables, initialement étudiées par M.O. Rabin sont closes par opérations booléennes, projection et produit cartésien (voir par exemple [Rab69],[Tho90]). La caractérisation précédente des forêts $\vec{O}$-reconnaissables permet de généraliser ces propriétés aux éléments de $\mathsf{TREC}^{\vec{O}}$.

Théorème 4.1.14. *Pour tout $\vec{O} = (O_1, \ldots, O_m)$, $O_i \subseteq A^*$, la famille $\mathsf{TREC}^{\vec{O}}$ est stable par union, intersection, complément, projection et produit carthésien.*

Preuve : La preuve est identique à celle des points 2 et 3 du théorème 4.1.8.
□

Étant donné $P \subseteq A^*$ un langage clos par préfixe, nous définissons une restriction de la classe $\mathsf{TREC}^{\vec{O}}$ permettant de "reconnaître" des forêts dans P-Arbre.

Définition 4.1.15 (Automate P-sécable). *Un automate $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}$ est P-sécable si il existe un état particulier $q_\perp \in Q$ tel que $c(q_\perp) = 0$ et*

$\forall t \in A^*\text{-}\mathsf{Arbre}(\Sigma)$, $r \in A^*\text{-}\mathsf{Arbre}(Q)$ *run de $\mathcal{A}$ sur t, pour tout $u \in A^*$:*

$$u \notin P \text{ ssi } r(u) = q_\perp$$

Dans ce cas, pour tout run r, les sous-arbres extérieurs à P sont acceptants et pour savoir si un run r est acceptant, il suffit de tester la condition sur les chemins infinis dans P (et donc si P est fini, tout run est acceptant). De plus, les sommets qui n'appartiennent pas à P sont marqués par l'étiquette $q_\perp$.

4.1.3 Forêts définissables

Le transfert des résultats de théorie des automates à la logique sur les arbres complets non étiquetés donné par Rabin peut être étendu aux automates à oracles. Nous établissons une correspondance bijective entre les forêts $\vec{O}$-reconnaissables et les modèles de SOM-formules sur l'arbre $\mathbf{T}(A)^{\vec{O}}$ (théorème 4.1.18). Il s'ensuit que le problème du vide pour les forêts $\vec{O}$-reconnaissables est équivalent avec le problème de la satisfiabilité des SOM-formules closes dans $\mathbf{T}(A)^{\vec{O}}$. Nous prouvons finalement une correspondance analogue entre les forêts dans $\mathsf{TREC}^{\vec{O}}{}_{|P}$ et les SOM-formules de $\mathbf{T}(A)_{|P}{}^{\vec{O}}$, où $P \subseteq A^*$ est clos par préfixe et SOM-définissable dans $\mathbf{T}(A)^{\vec{O}}$ (théorème 4.1.21).

Soit A un alphabet fini, et $Sig(A) = \{\varepsilon, (\mathrm{SUCC}_a)_{a\in A}\}$ une signature où SUCC_i sont des symboles relationnels binaires et ε symbolise une relation constante.

Soit ϕ une SOM$(Sig(A))$-formule à m places (voir définition 3.2.5), et n variables libres. Tout modèle $\vec{C}$ de Φ dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ correspond à un arbre dans P-$\mathsf{Arbre}(\{0,1\}^n)$: l'arbre caractéristique de $\vec{C}$ dans P.

Définition 4.1.16. *Une forêt $F \subseteq P\text{-}\mathsf{Arbre}(\{0,1\}^n)$ est dite* définissable *dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ si il existe une formule à $|\vec{O}|$ places et n variables libres $\phi(X_1, \ldots, X_n)$ définissant F au sens suivant :*

$$F = \{\mathrm{T}(A)^{(S_1,\ldots,S_n)}{}_{|P} \in P\text{-}\mathsf{Arbre}\{0,1\}^n : \mathbf{T}(A)_{|P}{}^{\vec{O}}, S_1, \ldots, S_n \models \phi(X_1, \ldots, X_n)\}.$$

Dans ce cas, nous notons (abusivement) $\mathbf{T}(A)_{|P}{}^{\vec{O}}, F \models \phi$.

Les forêts définissables dans un arbre complet non étiqueté sont bien connues, elles correspondent exactement aux forêts reconnaissables.

Théorème 4.1.17 (Rabin). *La classe des forêts reconnaissables $\mathsf{TREC}(A)$ correspond exactement à l'ensemble des forêts définissables dans $\mathbf{T}(A)$.*

La caractérisation des forêts $\vec{O}$-reconnaissables obtenue dans le lemme 4.1.12 permet de généraliser ce résultat aux arbres étiquetés.

Théorème 4.1.18. *Pour tout vecteur $\vec{O}$ de sous-ensembles de A^*, $\mathsf{TREC}(A)^{\vec{O}}$ correspond exactement à l'ensemble des forêts définissables en LSOM dans $\mathbf{T}(A)^{\vec{O}}$.*

Preuve : Soit $F \in \mathsf{TREC}^{\vec{O}}(A, \{0,1\}^n)$ avec $\vec{O} = (O_1, \dots O_m)$.

D'après le lemme 4.1.12, il existe une forêt reconnaissable $\widetilde{F} \in A^*\text{-}\mathsf{Arbre}(\{0,1\}^{n+m})$ telle que

$$F = \pi_{1,\dots,n}(\widetilde{F} \cap F_\chi(A)^{\vec{O}}(\{0,1\}^n)).$$

D'après le théorème 4.1.17, $\widetilde{F}$ est définissable dans $\mathbf{T}(A)$ et il existe donc une formule $\widetilde{\phi}$ ayant $n+m$ variables libres et telle que

$$\widetilde{F} = \{\mathrm{T}^{(S_1,\dots,S_{n+m})} : \mathbf{T}(A) \models \widetilde{\phi}(S_1, \dots, S_{n+m})\}.$$

Alors,

$$\begin{aligned} F &= \pi_{1,n}(\{\mathrm{T}^{(S_1,\dots,S_n,O_1,\dots,O_m)} : \mathbf{T}(A) \models \widetilde{\phi}(S_1, \dots, S_n, O_1, \dots, O_m)\}) \\ &= \{\mathrm{T}^{(S_1,\dots,S_n)} \mid \mathbf{T}(A) \models \widetilde{\phi}(S_1, \dots, S_n, O_1, \dots, O_m)\}. \end{aligned}$$

En remplaçant chaque variable X_{n+i} par le symbole relationnel O_i, nous transformons maintenant la formule $\widetilde{\phi}$ en une formule ϕ à m places et n variables libres. Cette formule vérifie

$$F = \{\mathrm{T}^{(S_1,\dots,S_n)} | \ \mathbf{T}(A)^{\vec{O}} \models \phi(S_1, \dots, S_n)\}$$

et F est donc définissable dans $\mathbf{T}(A)^{\vec{O}}$.

Inversement, si F est une forêt définissable par une formule $\phi(X_1, \dots, X_n)$, alors en appliquant la transformation inverse consistant à remplacer chaque symbole relationnel O_i par la variable Y_i, nous obtenons une formule $\widetilde{\phi}$ dont les variables libres sont $X_1, \dots, X_n, Y_1, \dots, Y_m$ et définissant une forêt $\widetilde{F} \in A^*\text{-}\mathsf{Arbre}\{0,1\}^{n+m}$ telle que

$$F = \pi_{1,\dots,n}(\widetilde{F} \cap F_\chi(A)^{\vec{O}}(\{0,1\}^n))$$

D'après le théorème 4.1.17, $\widetilde{F}$ est une forêt reconnaissable et F est donc $\vec{O}$-reconnaissable (lemme 4.1.12).

□

Corollaire 4.1.19. *Soit $\vec{O}$ un vecteur de langages dans A^*.*

Le problème du vide est décidable pour les forêts dans $\mathsf{TREC}^{\vec{O}}(A)$ ssi $\mathbf{T}(A)^{\vec{O}}$ admet une SOM-théorie décidable.

Preuve :

- Soit F une forêt $\vec{O}$-reconnaissable définie par une formule $\phi(X_1, \dots, X_n)$ et supposons que la SOM-théorie de $\mathbf{T}(A)^{\vec{O}}$ est décidable.
  Alors, comme $F \neq \emptyset$ ssi $\mathbf{T}(A)^{\vec{O}} \models \exists X_1, \dots, X_n.\ \phi(X_1, \dots, X_n)$, décider si F est vide revient à décider de la satisfiabilité d'une SOM-formule close.
- Inversement, supposons que pour toute forets $\vec{O}$-reconnaissable, le problème du vide est décidable. D'après le théorème 4.1.18, chaque formule close à m places ϕ sur $Sig(A)$ définit dans $\mathbf{T}(A)^{\vec{O}}$ une forêt $F \in A\text{-}\mathsf{Arbre}(\{0,1\}^0)$ telle que $F \neq \emptyset$ ssi $\mathbf{T}(A)^{\vec{O}} \models \phi$.

□

Rabin avait déjà prouvé ce résultat pour le cas $m = 0$, et montré par ailleurs que le problème du vide est décidable pour les forêts reconnaissables. Le théorème suivant est donc bien connu :

Théorème 4.1.20 ([Rab69]). *La théorie* SOM *de* $\mathbf{T}(A)$ *est décidable.*

Nous généralisons maintenant le théorème 4.1.18 aux forêts de domaine P, où P est un langage clos par préfixe :

Théorème 4.1.21. *Soit* $P \subseteq A^*$ *un langage clos par préfixe, et* $\vec{O}$ *un vecteur de sous-ensembles de* P *tel que* P *est SOM-définissable dans* $\mathbf{T}(A)^{\vec{O}}$.
La classe $\mathsf{TREC}^{\vec{O}}{}_{|P}$ *correspond exactement à l'ensemble des forêts définissables dans* $\mathbf{T}(A)_{|P}{}^{\vec{O}}$.

La première implication permettant de démontrer ce théorème est prouvé dans le lemme 4.1.22, et la seconde résulte des lemmes 4.1.23 et 4.1.24.

Lemme 4.1.22. *Soit* P *un langage préfixe appartenant à* $\mathsf{DEF}(\mathbf{T}(A)^{\vec{O}})$ *et* F *une forêt dans* P-*Arbre*(Σ). *Si* F *est définissable dans* $\mathbf{T}(A)_{|P}{}^{\vec{O}}$, *alors* $F \in \mathsf{TREC}^{\vec{O}}{}_{|P}$.

Preuve : Le langage P étant définissable dans $\mathbf{T}(A)$, pour toute SOM$(Sig(A))$-formule à m places $\phi(X_1, \ldots, X_n)$, il est possible, en relativisant la formule à P, de construire une nouvelle SOM$(Sig(A))$-formule à m places $\phi_P(X_1, \ldots, X_n)$ telle que $\forall S_1, \ldots, S_n \subseteq A^*$,

$$\mathbf{T}(A)^{\vec{O}} \models \phi_P(S_1, \ldots, S_n) \text{ ssi } \mathbf{T}(A)_{|P}{}^{\vec{O}} \models \phi(S_1, \ldots, S_n) \text{ et } S_1, \ldots, S_n \subseteq P$$

Si F est la forêt des modèles de ϕ dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$, et F' est la forêt des modèles de ϕ_P dans $\mathbf{T}(A)^{\vec{O}}$, alors F' correspond à la restriction à P de F' : $F = F'_{|P}$.
D'après le théorème 4.1.18 appliqué à ϕ_P, la forêt F' est $\vec{O}$-reconnaissable et donc $F \in \mathsf{TREC}^{\vec{O}}(P)$.
□

Pour l'implication inverse, nous commençons par nous restreindre à des forêts reconnues par automates P-sécables (voir définition 4.1.15).

Lemme 4.1.23. *Soit* $P \subseteq A^*$ *un langage clos par préfixe et* $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}(A)$ *un automate* P*-sécable. La forêt* $\mathrm{F}(\mathcal{A})_{|P}$ *est définissable dans* $\mathbf{T}(A)_{|P}{}^{\vec{O}}$.

Preuve : De façon évidente, la formule décrite dans la preuve du théorème 4.1.18 convient également dans le cas où $\mathcal{A}$ est P-sécable.
□

Pour achever la preuve du théorème 4.1.21, il reste à montrer que si chaque O_i est inclus dans P, alors tout automate dans $\mathsf{AFA}^{\vec{O}}(A)$ est équivalent (à l'intérieur de P) à un automate P-sécable dans $\mathsf{AFA}^{\vec{O}}(A)$.

Lemme 4.1.24. *Soit* $P \subseteq A^*$ *un langage clos par préfixe et* $\vec{O}$ *un vecteur de sous-ensembles de* P *tel que* P *est définissable en* LSOM *dans* $\mathbf{T}(A)^{\vec{O}}$. *Pour toute forêt* $F \in \mathsf{TREC}^{\vec{O}}(A)$, *il existe un automate* P*-sécable* $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}$ *tel que* $F_{|P} = \mathrm{F}(\mathcal{A})_{|P}$.

Preuve : Supposons pour simplifier que $A = \{a_0, a_1\}$, si A est de taille quelconque, la preuve est identique. Posons pour $i \in \{0, 1\}$, $P \cdot {a_i}^{-1} = \{u \in A^* | \, ua_i \in P\}$ et $\vec{P} = (P \cdot {a_0}^{-1}, P \cdot {a_1}^{-1})$. P étant définissable dans $\mathbf{T}(A)^{\vec{O}}$ par hypothèse, le vecteur $\vec{P}$ est également définissable dans $\mathbf{T}(A)^{\vec{O}}$ et d'après le théorème 4.1.18, il existe donc un automate $\mathcal{A}_P \in \mathsf{AFA}^{\vec{O}}(A, \{0, 1\}^2)$ tel que $\mathrm{F}(\mathcal{A}_P) = \{\mathrm{T}(A)^{\vec{P}}\}$.

Le théorème 4.1.14 assure de l'existence d'un automate $\mathcal{A}_1 \in \mathsf{AFA}^{\vec{O}}(A, \Sigma \times \{0, 1\}^2)$ vérifiant $\mathrm{F}(\mathcal{A}_1) = \mathrm{F}(\mathcal{A}) \hat{} \mathrm{F}(\mathcal{A}_P)$. Cet automate $\mathcal{A}_1$ permet donc de décrire $\mathrm{F}(\mathcal{A})$, ainsi que les *limites*

de P. Nous construisons maintenant à partir $\mathcal{A}_1$ un nouvel automate $\mathcal{B}$ capable de vérifier aux limites de P, qu'en continuant l'exécution de $\mathcal{A}$ en dehors de P, il serait possible de trouver un étiquetage des sous-arbres manquants tel que l'arbre ainsi complété admet un run acceptant dans $\mathcal{A}$. La condition $O_i \subseteq P$ assure que les seules transitions applicables en dehors de P ont pour test le vecteur nul $(0,\ldots,0)$. Ainsi, sur les sommets extérieurs à P, l'automate se comporte comme un automate sans oracles.

Pour tout $q \in Q_1$ (l'ensemble des états de $\mathcal{A}_1$), nous construisons $\mathcal{A}_q \in \mathsf{AFA}^{\vec{O}}$ à partir de l'automate $\mathcal{A}_1$ en retirant toutes les transitions à test non nul et en posant q comme état initial. Sur les mots n'appartenant pas à P, cet automate est équivalent à un automate sans oracles puisque le vecteur caractéristique de tout $u \notin P$ est constant et égal à $(0,\ldots,0)$. D'après le théorème de Rabin 4.1.20, nous savons effectivement décider si $L(\mathcal{A}_q) \neq \emptyset$ et donc construire l'ensemble $Acc = \{q \in Q_1 : L(\mathcal{A}_q) \neq \emptyset\}$.

Ainsi, Acc décrit les états à partir desquels, en dehors de P, il existe un sous-arbre acceptant. Donc, $\forall t$ et r, tels que r est un run de $\mathcal{A}_1$ sur t, $\forall u \in P$ avec $t(u) = (\alpha, b_0, b_1)$, si il existe une direction $i \in \{0,1\}$ telle que $b_i = 0$, alors $ua_i \notin P$. À partir de l'ensemble Acc, il est donc possible de construire un automate P-sécable $\mathcal{B}_1$, tel que $\mathrm{F}(\mathcal{A}_1)_{|P\times\{0,1\}} = \mathrm{F}(\mathcal{B}_1)_{|P\times\{0,1\}}$. Il suffit pour cela de modifier l'ensemble des transitions de $\mathcal{A}_1$ de la façon suivante :

- remplaçons chaque transition $(p, (\alpha, b_0, b_1), \vec{o}, p_0, p_1)$ pour $r(u) \in Acc$, par la même transition dans laquelle, pour chaque i tel que $b_i = 0$ l'état p_i a été substitué par $q_\perp$. Par définition de Acc et puisque $b_i = 0$ indique que la direction i va faire sortir de P, cette substitution ne modifie pas la restriction à P de la forêt reconnue par l'automate $\mathcal{A}_1$.
- supprimons chaque transition $(p, (\alpha, b_0, b_1), \vec{o}, p_0, p_1)$ telle que $r(u) \notin Acc$. Puisque cette transition n'amènera jamais à un run acceptant, sa suppression ne modifie pas le langage accepté par l'automate $\mathcal{A}_1$.

À partir de ce nouvel automate, il est alors facile de construire un automate P-sécable $\mathcal{B}$ reconnaissant la forêt $\mathrm{F}(\mathcal{B}_1)$. Il suffit pour cela de choisir comme ensemble d'état $Q = Q_{\mathcal{B}_1} \times \{0,1\}^2 \cup \{q_\perp\}$, et de modifier les transitions de $\mathcal{B}_1$ de façon déplacer dans les états les vérifications faites sur les noeuds des arbres. Nous avons alors $\mathrm{F}(\mathcal{B})_{|P} = \mathrm{F}(\mathcal{A})_{|P}$.

□

Remarque 4.1.25. *Le théorème reste valable si les O_i sont non inclus dans P mais sont tels que l'application $u \mapsto \chi_{\vec{O}}(u)$ est constante en dehors de P.*

4.2 Arbres régulier et problème du vide

Les arbres réguliers forment une famille remarquable correspondant à celle des dépliages des graphes finis. Ces arbres sont utilisés dans de nombreux domaines de l'informatique (voir [Cou83] pour un survol). M.O. Rabin prouve que toute forêt reconnaissable contient un arbre régulier et que le problème du vide est décidable pour les forêts reconnaissables. Ces résultats associés à la représentation de toute forêt reconnaissable par une SOM-formule (théorème 4.1.17) prouvent le théorème 4.1.20 formulant la décidabilité de la SOM-théorie de l'arbre complet $\mathbf{T}(A)$. Tous ces résultats sont prouvés dans [Rab69] et dans [Tho90].

Nous généralisons ici la notion d'arbre régulier en définissant une classe d'arbre correspondant aux dépliages de graphes de calculs d'automates de mots à oracles. Nous discutons ensuite des liens entre l'existence d'un tel arbre dans une forêt $\vec{O}$-reconnaissable et la propriété du Modèle Définissable (définition 3.1.2) pour l'arbre $\mathbf{T}(A)^{\vec{O}}$. Enfin, nous terminons cette section

en définissant les automate d'arbres *sans entrées*. Nous montrons que l'étude du problème du vide et de l'existence d'un arbre régulier pour les forêts $\vec{O}$ reconnaissable peut être réduite à l"étude des forêts engendrés par de tels automates, ce qui permettra de simplifier les preuves de la section suivante.

4.2.1 Arbre réguliers

Un arbre $t \in \mathsf{Arbre}(\Sigma)$ est dit **$\vec{O}$-régulier** si il existe un automate de mot déterministe $\mathcal{A} \in \mathsf{AF}^{\vec{O}}$ et une fonction $out : Q \to \Sigma$ *générant* t, i.e., tels que $\forall u \in \{0,1\}^*$,

$$(q_{0,\uparrow}\, u) \to_{\mathcal{A}} (q, u_{\uparrow}) \text{ ssi } out(q) = t(u)$$

Remarque 4.2.1.

1. *Un arbre est régulier ssi il est $\vec{\emptyset}$-régulier.*
2. *Soit t un arbre régulier généré par un automate déterministe $\mathcal{A} \in \mathsf{AF}^{\vec{O}}(A, \Sigma)$, alors pour tout $\alpha \in \Sigma$, l'ensemble $L_\alpha = \{u |\ t(u) = \alpha\}$ est $\vec{O}$-régulier.*
3. *Tout arbre $\vec{O}$-régulier sur A^* est isomorphe à la restriction à $L_\chi^{\vec{O}}$ (le langage caractéristique de $\vec{O}$ (définition 4.1.4)) d'un $(A \times \{0,1\}^m)^*$-arbre.*
4. *Pour tout $\vec{O}$, l'arbre caractéristique $\mathrm{T}^{\vec{O}}$ est $\vec{O}$-régulier.*

Nous étendons cette définition aux P-arbres en disant que $t \in P\text{-}\mathsf{Arbre}(\Sigma)$ est $\vec{O}$-régulier lorsqu'il existe un arbre $\vec{O}$-régulier $t' \in A^*\text{-}\mathsf{Arbre}(\Sigma)$, tel que $t = t'_{|P}$.

Théorème 4.2.2 ([Rab69]).

1. *Toute forêt reconnaissable $F \in A^*\text{-}\mathsf{Arbre}(\Sigma)$ contient un arbre régulier.*
2. *Le problème du vide est décidable pour les forêts reconnaissables.*

4.2.2 Propriété du Modèle Définissable

Nous utilisons maintenant les liens démontrés précédemment entre forêt $\vec{O}$-reconnaissables et SOM-formules dans la structure $\mathbf{T}(A)^{\vec{O}}$ pour traduire en termes de forêts et d'arbres réguliers, la propriété du Modèle définissable (MD) donnée dans la définition 3.1.2.

Proposition 4.2.3. *Si P est* SOM*-définissable dans $\mathbf{T}(A)^{\vec{O}}$, les propriétés suivantes sont équivalentes :*

1. $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ *vérifie la propriété* MD
2. *pour toute forêt $\vec{O}$-reconnaissable non vide $F \subseteq P\text{-}\mathsf{Arbre}(\{0,1\})$, il existe $\vec{D}$,* SOM*-définissable dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$, tel que F contient un arbre $\vec{D}$-régulier.*

Preuve : Supposons que P est SOM-définissable dans $\mathbf{T}(A)^{\vec{O}}$, une simple réécriture de la propriété MD utilisant le théorème 4.1.21 implique l'équivalence des deux propriétés suivantes :

- $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ vérifie MD
- pour toute forêt $\vec{O}$-reconnaissable non vide $F \subseteq P\text{-}\mathsf{Arbre}(\{0,1\})$, il existe un ensemble D, SOM-définissable dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$, tel que l'arbre $\mathrm{T}(A)^D{}_{|P}$ appartient à F.

1. Supposons que $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ vérifie MD. Toute forêt $\vec{O}$-reconnaissable contient donc un arbre $\mathrm{T}(A)^{\vec{O}}{}_{|P}$, où D est SOM-définissable dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$. Cet arbre est donc D-régulier.
2. Inversement, supposons $F \subseteq P\text{-}\mathsf{Arbre}(\{0,1\})$ est une forêt $\vec{O}$-reconnaissable contenant un arbre t qui est $\vec{D}$-régulier, où D est SOM-définissable dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$. Par définition d'arbre régulier, la forêt $\{t\}$ est donc $\vec{D}$-reconnaissable. Soit $S \subseteq P$ tel que $t = \mathrm{T}^S{}_{|P}$, le théorème 4.1.21, assure que S est SOM-définissable dans $\mathbf{T}(A)_{|P}{}^{\vec{S}}$. Puisque $\vec{D}$ est SOM-définissable dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$, S l'est également. D'après l'équivalence établie au début de la preuve, $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ vérifie MD.

□

Le second point du théorème 4.2.2 se réécrit donc de la façon suivante :

Théorème 4.2.4. *Pour tout alphabet fini A, la structure $\mathbf{T}(A)$ vérifie la propriété* MD.

4.2.3 Automate d'arbre sans entrée

Pour traiter du problème du vide ou de l'existence d'un arbre régulier, il est possible de se restreindre à des automates **sans entrée** i.e., tels que l'alphabet d'entrée Σ est vide. Dans un tel automate, la relation de transition est donc $\Delta \subseteq Q \times \{0,1\} \times Q^n$.

Tout automate $\mathcal{A} = (Q, \Sigma, A, \vec{O}, \Delta, q_0, c) \in \mathsf{AFA}^{\vec{O}}$ peut être transformé en un nouvel automate $\mathcal{B} = (Q \times \Sigma, \emptyset, A, \vec{O}\Delta', Q_0, c') \in \mathsf{AFA}^{\vec{O}}$ sans entrée où Δ' contient une transition $((q,\alpha), \vec{o}, (p_1,\alpha_1), \ldots, (p_n,\alpha_n))$ ssi $(q, \alpha, \vec{o}, p_1, \ldots, p_n) \in \Delta$. Q_0 contient tous les (q_0, α), $\alpha \in \Sigma$ et $c'(q,\alpha) = c(q)$, $\forall \alpha \in \Sigma$. (Il reste encore à réduire Q_0 à un seul état, cette construction étant classique, nous ne la détaillons pas ici.)

Clairement, pour tout $\vec{O}$, les runs acceptants de $\mathcal{B}$ sont exactement les paires $r' = r \hat{} t$ où r est un run acceptant de $\mathcal{A}$ sur t. Donc pour tout vecteur $\vec{R}$, t est $\vec{R}$-régulier ssi r' est $\vec{R}$-régulier et $L(\mathcal{A}) = \emptyset$ ssi $L(\mathcal{B}) = \emptyset$.

Le lemme suivant est donc immédiat et permettra de restreindre nos prochaines preuves à des automates sans entrées.

Proposition 4.2.5. *Soit $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}$, il existe un automate sans entrée $\mathcal{B} \in \mathsf{AFA}^{\vec{O}}$ tel que :*

1. *$\mathrm{F}(\mathcal{A}) = \emptyset$ ssi $\mathcal{B}$ admet un run acceptant,*
2. *pour tout vecteur $\vec{R}$ de sous-ensembles de A^*, $\mathrm{F}(\mathcal{A})$ contient un arbre $\vec{R}$-régulier ssi $\mathcal{B}$ admet un run $\vec{R}$-régulier,*
3. *soit $P \subseteq A^*$ un langage clos par préfixe, si $\mathcal{A}$ est P-sécable, alors $\mathcal{B}$ est P-sécable.*

Un automate sans entrée déterministe $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}$ admet au plus un run. Si il existe, ce run est un arbre $\vec{O}$-régulier.

Proposition 4.2.6. *Pour tout $\vec{O}$, pour tout $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}$ sans entrée et déterministe, si il existe un run de $\mathcal{A}$ alors ce run est $\vec{O}$-régulier.*

Preuve : Soit $\mathcal{A} = (Q, A, \vec{O}, \Delta, q_0, c) \in \mathsf{AFA}^{\vec{O}}$ un automate sans entrée déterministe. nous construisons $\mathcal{A}_r = (Q, A, \vec{O}, \Delta_r, q_0) \in \mathsf{AF}^{\vec{O}}$ déterministe où Δ_r est composé de toutes les transitions $(q, a_i, \vec{o}, p_i)$, $i \in [1,n]$ telles que $(q, \vec{o}, p_1, \ldots, p_n) \in \Delta$. Si t est un run acceptant de $\mathcal{A}$, alors t est l'arbre engendré par $\mathcal{A}_r$ associé à la fonction $out : q \mapsto q$, $\forall q \in Q$.

□

4.3 Logique pour automate à oracles restreints

Afin de construire des structures ayant de "bonnes propriétés", nous nous restreignons maintenant à des oracles de la forme $\vec{O} = (\mu_{|P}{}^{-1}(R_1), \ldots, \mu_{|P}{}^{-1}(R_m))$, où μ est un morphisme de A^* dans un monoïde et P est un sous-ensemble de A^* clos par préfixe.

Nous utilisons la théorie des jeux pour exprimer certains problèmes relatifs aux forêts reconnaissables par des formules SOM sur la structure image $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$. Ceci permet de prouver que si P est SOM-définissable dans $\mathbf{T}(A)^{\vec{O}}$, alors la propriété "avoir une SOM-théorie décidable" peut-être transférée de $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ à $\mathbf{T}(A)_{|P}{}^{\vec{O}}$.

Nous définissons également une condition sur $\mu_{|P}$ rendant possible le transfert de la propriété MD, et permettant de prouver l'égalité entre la classe des ensembles SOM-définissables dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ et l'union des classes $\mathsf{REC}^{\mu_{|P}{}^{-1}(\vec{D})}$ pour $\vec{D}$ vecteur SOM-définissable dans la structure $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$.

4.3.1 Jeux de parité

Nous considérons des jeux à deux joueurs (le joueur 0 et le joueur 1) constitués d'un graphe dont les sommets sont partitionnés en positions du joueur 0 et du joueur 1, d'une position de départ et d'une condition de gain.

Au début d'une partie, un pion est déposé sur un des sommets du graphe. Le joueur ayant le pion sur une de ses positions le déplace le long des arcs du graphe jusqu'à arriver sur une position du joueur adverse. C'est alors à celui-ci de jouer, il itère le même procédé. Si un joueur ne peut plus jouer, (i.e., si il n'y a pas d'arcs issus du sommet sur lequel le pion est posé) il est déclaré perdant. Dans le cas contraire la partie est infinie et correspond à un chemin infini dans le graphe. Le gagnant est alors déterminé en testant la condition de gain sur la suite infinie de sommets formée par la partie.

Nous nous intéresserons en particulier aux jeux de parité (voir par exemple [GH82, EJ91, GTW02]) :

Définition 4.3.1 (Jeu de parité). *Un jeu de parité est un tuple $J = (V_0, V_1, E, v_0, c)$ où $V = V_0 \uplus V_1$ est un ensemble de sommets partitionné en sommets du joueur 0 et sommets du joueur 1, (V, E) est un graphe, $v_0 \in V$ est le départ du jeu et $c : V \to [0, max]$ est une application associant à chaque sommet du graphe une priorité sous forme d'un nombre entier pris dans un intervalle borné.*

Une **partie** *dans J est un chemin issu de v_0 (fini ou infini) dans le graphe (V, E).*

Si la partie est finie et termine sur un sommet $v \in V_\epsilon$, $\epsilon \in \{0, 1\}$ (i.e., si il n'y a pas d'arcs sortant de v), le joueur ϵ est déclaré perdant (et par suite l'autre est gagnant). Sinon, le gagnant est déterminé par n_0, valeur de la plus petite priorité rencontrée infiniment souvent dans la partie. En d'autres mots, si la partie est $v_0 v_1 \cdots v_n \cdots$ alors n_0 est le plus petit entier ayant une infinité d'occurrences dans le mot $c(v_0)c(v_1)\cdots c(v_n)\cdots$. Si n_0 est pair, le joueur 0 est déclaré gagnant, dans le cas contraire, c'est 1 qui gagne.

Une **stratégie** pour le joueur ϵ ($\epsilon \in \{0, 1\}$) est une fonction s associant à tout préfixe d'une partie $\pi = v_0 v_1 \cdots v_n$ tel que $v_n \in V_\epsilon$, un sommet v_{n+1} tel que $(v_n, v_{n+1}) \in E$. Une stratégie

est dite **positionnelle** si elle ne dépend que du sommet courant v_n. Dans ce cas, nous la représentons par une application de V_ϵ dans V. Une partie $\pi = v_0 v_1 \cdots v_n \cdots$ est **conforme** à s si pour tout $i \geq 0$ tel que $v_i \in V_\epsilon$, alors $v_{i+1} = \mathrm{s}(v_0 \dots v_i)$. Une stratégie s est **gagnante** pour le joueur ϵ si toute partie conforme à s est gagnante pour le joueur ϵ.

La notion de stratégie gagnante permet de capturer les sommets à partir desquels un joueur est sûr de gagner (si il joue bien). Le joueur ϵ **gagne** le jeu si il existe une stratégie **gagnante**, i.e., telle que toute partie conforme à s est gagnante.

Le résultat suivant est fréquemment utilisé pour l'application de la théorie des jeux à la théorie des modèles. Il est prouvé dans [EJ91, GTW02].

Théorème 4.3.2. *Pour tout jeu de parité J :*

1. *un et seul joueur admet une stratégie gagnante dans J,*
2. *si le joueur ϵ gagne le jeu, alors il admet une stratégie positionnelle gagnante.*

Pour éviter le cas particulier des parties finies, il est commode de se restreindre à des jeux dits *complets*.

Définition 4.3.3 (Jeu complet). *Un jeu est dit* **complet** *si il ne n'admet aucune partie finie.*

Lemme 4.3.4. *Pour tout jeu de parité J, il existe un jeu complet J' tel que*
- *le joueur 0 gagne J ssi il gagne J',*
- *étant donnée une stratégie pour joueur 0 gagnante dans J', il existe un algorithme permettant de calculer une stratégie pour joueur 0 gagnante dans J.*

Preuve : Soit $J = (V_0, V_1, E, c, v_0)$. Nous construisons le jeu complet $J^c = (V_0^c, V_1^c, E^c, c, v_0)$ où
- $V_0^c = \{v_\perp\} \cup V_0$
- $V_1^c = \{v_\top\} \cup V_1$
- c est étendue aux deux nouveaux sommets en posant $c(v_\perp) = max$ et $c(v_\top) = 0$
- $E^c = E \cup \{(v_\perp, v_\perp), (v_\top, v_\top)\} \cup \{(v, q_\perp) \mid v \in V_0, (v \times V) \cap E = \emptyset\} \cup \{(v, q_\top) \mid v \in V_1, (v \times V) \cap E = \emptyset\}$

Si s est une stratégie gagnante dans J^c pour 0, s définit également une stratégie gagnante dans J pour 0.

□

Pour étudier les parties jouées selon une stratégie positionnelle, il est suffisant de considérer le nouveau jeu obtenu en restreignant le graphe du jeu initial aux parties conformes à la stratégie.

Définition 4.3.5 (Jeu restreint à une stratégie). *Soit $J = (V_0, V_1, E, c, v_0)$ un jeu et* s *une stratégie positionnelle pour le joueur 0. Nous associons à J et* s *le jeu restreint $J_\mathrm{s} = (V_0 \cup V_1, \emptyset, E^\mathrm{s}, c, v_0)$ où $E^\mathrm{s} = E_1 \cup \{(v, v') \mid v \in V_0, \mathrm{s}(v) = v'\}$.*

Lemme 4.3.6. *Pour tout jeu $J \in \mathsf{Jeu}(\mathfrak{I})$ sans parties finies, pour toute stratégie positionnelle* s*, la stratégie* s *est gagnante dans J ssi le jeu réduit J_s est gagnant pour 0.*

Preuve : Par définition de jeu restreint, une suite infinie de sommets $v_0 \cdots v_n \cdots$ définit une partie dans J_s ssi elle définit une partie dans J conforme à s. Puisque les priorités des sommets des deux jeux sont les mêmes, nous en déduisons que toute partie jouée selon s dans J est gagnante pour 0 ssi toute partie dans J_s est gagnante pour le joueur 0.

□

Remarquons que ce résultat n'est plus valable si il existe dans J des parties finies terminant en V_0. En effet, une telle partie est perdante dans J pour joueur 0, alors qu'elle est perdante pour 1 dans J^{s}.

Introduisons maintenant la logique sur les jeux. Tout jeu $J = (V_0, V_1, E, v_0, c)$ peut être associé à une structure relationnelle

$$\mathsf{J} = \langle V, V_0, V_1, E, v_0, c_0, \ldots, c_{max} \rangle$$

où pour tout $i \in [0, max]$, $c_i = \{v \in V \mid c(v) = i\}$.

Pour toute stratégie sans mémoire S du joueur 0, le problème de savoir si S est gagnante se réduit à la satisfiabilité d'une SOM-formule close dans le graphe de calcul de la machine associée au jeu réduit à S.

Lemme 4.3.7. *Pour tout jeu complet J, il existe une SOM-formule close* SG *telle que :* $\forall$S *stratégie sans mémoire pour joueur 0 dans J,*

$$\mathsf{J}_{\mathrm{S}} \models \mathrm{SG} \textit{ ssi } \mathrm{S} \textit{ est gagnante}$$

Preuve : Commençons par construire une formule P_0 telle que $\mathsf{J}_{\mathrm{S}} \models P_0$ ssi il existe une partie dans J^{s} perdue par joueur 0, i.e., ssi il existe un chemin infini $v_0 \cdots v_n \cdots$ tel que le plus petit entier apparaissant infiniment souvent dans $c'(v_0) \cdots c'(v_n) \cdots$ est impair.
$P_0 := \exists X, \exists y_0 \in X$.

1. $\exists n_0$ impair tel que $c_{n_0}(y_0)$
2. $n_0 = min\{n |\ c_n(y),\ y \in X\}$
3. $v_0 \overset{*}{\rightarrow} y_0$
4. $\forall y \in X, \exists y'$ tel que $c_{n_0}(y')$ et $y \overset{+}{\rightarrow} y'$

D'après le théorème 4.3.2, le joueur 0 perd le jeu ssi joueur 1 gagne le jeu, donc $\mathrm{SG} := \neg P_0$.

□

4.3.2 Jeux d'accessibilité pour automates à oracles

Nous utilisons maintenant les jeux de parité afin d'exprimer des problèmes liés aux forêts $\vec{O}$-reconnaissable. Nous fixons pour toute cette sous-section les objets suivants :

- A est un alphabet fini, que nous considérons pour simplifier réduit à deux éléments : $A = \{a_0, a_1\}$ (tous les résultats établis restent valides pour A quelconque),
- μ est un morphisme surjectif de A^* vers un semi-groupe $(M, \star)$,
- P est un sous-ensemble de A^* clos par préfixe,
- $\mathcal{O}_{\mu_{|P}}$ désigne l'ensemble des vecteurs $\vec{O} = ({\mu_{|P}}^{-1}(R_1), \ldots, {\mu_{|P}}^{-1}(R_m))$ pour $R_i \subseteq M$.

Rappelons que d'après la définition 3.2.2, la structure *image par* μ de $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ est

$$\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}}) = \langle \mu(P),\ \mu(\varepsilon),\ \star_{\mu(a_0)} \star_{\mu(a_1)},\ \vec{R} \rangle,$$

où $\star_{\mu(a_i)} = \{(\sigma, \sigma \star \mu(a_i))\}$, pour $i \in \{1, 2\}$.

Nous prouvons que pour tout $\vec{O} \in \mathcal{O}_{\mu_{|P}}$, le problème du vide pour les forêts $\vec{O}$-reconnaissables se réduit à déterminer le gagnant d'un jeu de parité se déroulant dans une arène dont les sommets sont inclus dans le produit de $\mu(P)$ avec un ensemble fini. Nous en déduisons (proposition 4.3.15(1)) que pour ces forêts, le problèmes du vide se réduit à celui de la satisfiabilité d'une SOM-formule close dans la structure $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$, image par μ de la structure $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ (voir définition 3.2.2). Nous prouvons également que toute forêt $\vec{O}$-reconnaissable non vide contient un arbre ${\mu_{|P}}^{-1}(\vec{D})$-régulier, où $\vec{D}$ est un vecteur SOM-définissable dans $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ (proposition 4.3.15.2).

Nous restreignons tout d'abord notre étude aux automates d'arbres sans entrée et P-sécable. L'avantage de travailler avec de tels automates est que la validé d'un run de dépend que des sommets appartenant à P. De plus, les éléments qui n'appartiennent pas à P sont indiqués par l'étiquette $q_\perp$ et sont donc facilement identifiables.

Définition 4.3.8 (Jeu à oracles). *Associons à tout automate complet et P-sécable $\mathcal{A} = (Q, \emptyset, A, \vec{O}, \Delta, q_0, c)$, $\vec{O} \in \mathcal{O}_{\mu_{|P}}$, le jeu de parité suivant :*

$J_\mathcal{A} = (V_0, V_1, E, c', (\mu(\varepsilon), q_0))$ où
$V_0 = \mu(P) \times Q$ et $V_1 = \mu(P) \times \Delta$,
$E = E_0 \cup E_1$ où $E_0 \subseteq V_0 \times V_1$, $E_1 \subseteq V_1 \times V_0$ et
$E_0 = \{((\sigma, p), (\sigma, \delta)) |\ \delta = (p, \chi_{\mu(\vec{O})}(\sigma), p_0, p_1) \in \Delta\}$,
$E_1 = \{((\sigma, \delta), (\sigma \star \mu(i), p_i)) |\ p_i \neq q_\perp,\ (p, \vec{o}, p_0, p_1) \in \Delta,\ i \in \{0, 1\}\}$ et
c' est défini par $c'(\sigma, p) = c(p)$ et $c'(\sigma, (p, \vec{o}, p_0, p_1)) = c(p)$.

Une partie débute à la position $(\mu(\varepsilon), q_0)$. Chaque joueur joue alternativement en avançant dans le jeu. En position (σ, p), le joueur 0 choisit une transition $\delta = (p, \vec{o}, p_0, p_1)$ parmi celles vérifiant $\vec{o} = \chi_{\mu(\vec{O})}(\sigma)$ et se déplace en (u, δ). J1 choisit alors la direction i qu'il va suivre ($i \in \{0, 1\}$) et va en position $(\sigma \star \mu(a_i), p_i)$. Ainsi, puisque μ est un morphisme, pour tout préfixe de partie terminant en $(\sigma, x) \in V$, si u est le mot formé par la suite des directions choisies par le joueur 1, alors $\mu(u) = \sigma$. L'automate ayant été choisi complet, le jeu est également complet.

Pour un tel jeu, il est plus commode de voir une stratégie pour le joueur 0 comme une application des préfixes de parties dans Δ (ainsi, une stratégie positionnelle est une application $\mathrm{s} : V_0 \to \Delta$).

Tout run acceptant de $\mathcal{A}$ décrit une stratégie gagnante pour le joueur 0, puisque la suite des transitions choisies durant la partie est fixée par le run, et que par définition d'automate P-sécable, les chemins manquants dans le jeu correspondent à des sous-arbres acceptants étiquetés par $q_\perp$. Inversement, une stratégie gagnante s pour le joueur 0 donne une méthode pour construire un run acceptant de $\mathcal{A}$. Il suffit pour cela d'appliquer successivement les transitions indiquées par la stratégie.

Lemme 4.3.9. *Un automate sans entrée et P-sécable $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}$, $\vec{O} \in \mathcal{O}_{\mu_{|P}}$ admet un run acceptant ssi le joueur 0 admet une stratégie gagnante dans $J_\mathcal{A}$.*

Preuve :

- Soit r est un run de $\mathcal{A}$, nous obtenons la stratégie sans mémoire associée s_r en posant : $\forall v_0 \dots v_n$ préfixe d'une partie avec $v_n \in V_0$ et tel que la suite des directions prises par le joueur 1 est $u \in A^*$,

$$\mathrm{s}_r(v_0 \dots v_n) = (r(u), \chi_{\vec{O}}(u), r(ua_0), r(ua_1))$$

  Clairement, s_r est gagnante ssi r est acceptant.

- Soit s une stratégie pour le joueur 0, à chaque sommet u, nous appliquons la transition indiquée par $\mathrm{s}(\rho_u)$, où ρ_u est le préfixe de la partie jouée selon s, dont la suite des directions choisies par le joueur 1 est u :
  - $r_{\mathrm{s}}(\varepsilon) = q_0$ et $\rho_\varepsilon = (\mu(\varepsilon), q_0)$,
  - $\forall u \in P$, si $\mathrm{s}(\rho_u) = \delta$, avec $\delta = (p, \vec{o}, p_0, p_1)$, alors $\forall i \in \{0,1\}$ tel que $p_i \neq q_\perp$, $r_{\mathrm{s}}(ua_i) = p_i$ et $\rho_{ua_i} = \rho_u \cdot \mathrm{s}(\rho_u) \cdot (\mu(ua_i), p_i)$,
  - si $u \notin P$, $r_{\mathrm{s}}(u) = q_\perp$.

  L'arbre ainsi construit est clairement un run de $\mathcal{A}$ et il est acceptant ssi s est gagnante.

□

En utilisant le théorème 4.3.2, ce résultat se réécrit de la façon suivante :

Lemme 4.3.10. *Un automate sans entrée et P-sécable $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}$, $\vec{O} \in \mathcal{O}_{\mu_{|P}}$ admet un run acceptant ssi le joueur 0 admet une stratégie* **positionnelle** *gagnante dans $J_{\mathcal{A}}$.*

Soit s une stratégie positionnelle pour le joueur 0. Nous avons prouvé dans le lemme 4.3.7 que le problème "s est-elle une stratégie gagnante ?" se réduit au problème de la validité d'une formule close dans la structure associé au jeu restreint à s. Nous réduisons maintenant ce second problème à celui de la satisfiabilité d'une formule close dans la structure $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ augmentée d'un vecteur de relations unaires codant s. Pour cela, nous représentons chaque sous-ensemble de V, et chaque stratégie positionnelle pour 0, par un vecteur de sous-ensembles de $\mu(P)$.

Soit $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}$, $\vec{O} \in \mathcal{O}_{\mu_{|P}}$, nous supposons :

- $\Delta = \{\delta_1, \ldots, \delta_d\}$, où $\delta_i \neq \delta_j$ pour tout $i \neq j$, (dans ce cas, nous dirons que $\mathcal{A}$ est de *taille* d)
- $Q = \{s_1, \ldots, s_\tau\}$.

Nous définissons alors les applications suivantes :

- pour tout $D \subseteq V$, $g(D) = (g_1(D), \ldots, g_d(D), h_1(D), \ldots, h_\tau(D))$, où $\forall i \in [1,d]$, $j \in [1,\tau]$,

$$g_i(D) = \{\sigma \mid (\sigma, \delta_i) \in D\} \text{ et } h_j(D) = \{\sigma \mid (\sigma, s_j) \in D\},$$

- nous associons à tout stratégie positionnelle s pour le joueur 0 dans J le vecteur $\vec{S_{\mathrm{s}}} = (S_{\mathrm{s},1}, \ldots, S_{\mathrm{s},d})$, $S_{\mathrm{s},i} \subseteq \mu(P)$ où $\forall i \in [1, \ldots d]$,

$$S_{\mathrm{s},i} = \{\sigma \in \mu(P) \mid \mathrm{s}(\sigma, \pi_1(\delta_i)) = \delta_i\}.$$

Remarquons que le vecteur $\vec{S_{\mathrm{s}}}$ caractérise complètement s.

Lemme 4.3.11. *Soit $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}$ un automate sans-entrée et P-sécable avec $\vec{O} \in \mathcal{O}_{\mu_{|P}}$, et $\phi(X_1, \ldots, X_n)$ une SOM-formule sur $J_{\mathcal{A},\mathrm{s}}$, il existe une formule ϕ^g à d places sur $\mu(\mathbf{T}(A)_{|P})$ telle que $\forall$s stratégie positionnelle pour le joueur 0 dans J, $\forall D_1, \ldots D_\ell \subseteq V$*

$$J_{\mathcal{A},\mathrm{s}} \models \phi(D_1, \ldots, D_n) \text{ ssi } \mu(\mathbf{T}(A)_{|P})^{\vec{S_{\mathrm{s}}}} \models \phi^g(g(D_1), \ldots, g(D_\ell))$$

Preuve : Construisons ϕ^g dans le cas où ϕ est une formule atomique :

- Pour tous $i \in [1,d]$, $j \in [1,\tau]$, $\sigma, \sigma' \in M$, nous avons :
  - $E_{\mathrm{s}}((\sigma, \delta_i), (\sigma', p_j))$ ssi il existe une direction $\epsilon \in \{0,1\}$ telle que $\sigma' = \sigma \star \mu(a_\epsilon)$ et $\pi_{2+\epsilon}(\delta_i) = p_j$
  - $E_{\mathrm{s}}((\sigma, p_j), (\sigma', \delta_j))$ ssi $\sigma = \sigma'$ et $\pi_1(\delta_i) = p_j$ et $\sigma \in S_{\mathrm{s},i}$

  La formule E^g_{s} est donc clairement exprimable dans $\mu(\mathbf{T}(A)_{|P})^{\vec{S_{\mathrm{s}}}}$.

- $c^g_{\mathrm{s},n}(X_1,\ldots,X_{p+\tau})$ correspond à la disjonction des deux formules suivantes :
  - $\exists i \in [1,d]$ tel que $X_i = \{x\}$ et les autres sont vides et $c(\delta_i) = n$,
  - $\exists j \in [1,\tau]$ tel que $X_{d+j} = \{x\}$ et les autres sont vides et $c(p_j) = n$.
- si $\phi(X,Y) := X \subseteq Y$,
  alors $\phi^g(X_1,\ldots,X_{d+\tau},Y_1,\ldots,Y_{d+\tau}) := \forall i \in [1,d+\tau], X_i \subseteq Y_i$.

Finalement, si ϕ n'est pas une formule atomique, ϕ^g est donnée par une induction évidente.

□

En composant le lemme 4.3.11 et le lemme 4.3.7, nous déduisons immédiatement le résultat suivant :

Lemme 4.3.12. *Soit $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}$ un automate sans-entrée et P-sécable de taille d, avec $\vec{O} \in \mathcal{O}_{\mu_{|P}}$, il existe une SOM-formule close à d place,* SG *sur $\mu(\mathbf{T}(A)_{|P})$, telle que pour toute stratégie positionnelle* S *pour le joueur 0 dans $J_{\mathcal{A}}$,*

$$\mu(\mathbf{T}(A)_{|P})^{\vec{S}_{\mathrm{s}}} \models \mathrm{SG} \textit{ ssi } \mathrm{S} \textit{ est gagnante}$$

Nous montrons maintenant que la propriété "il existe une stratégie gagnante pour le joueur 0 dans $J_{\mathcal{A}}$" se réduit au problème de la satisfiabilité d'une formule close dans la structure $\mu(\mathbf{T}(A)_{|P})^{\vec{O}}$. En utilisant le lemme précédant, il suffit de vérifier que pour tout vecteur $\vec{S}$ de d sous-ensembles de $\mu(P)$, la propriété "$\vec{S}$ code une stratégie pour le joueur 0" est exprimable dans $\mu(\mathbf{T}(A)_{|P})^{\vec{O}}$. En supposant $\mathcal{A}$ complet (i.e., pour tout $(p,\vec{o})$, il existe une transition $(p,\vec{o},\vec{p})$), il suffit juste de s'assurer que pour tout état p, pour tout $\sigma \in \mu(P)$, il existe un et un seul $i \in [1,d]$ tel que

- la première composante de δ_i est p,
- $\chi_{\mu(\vec{O})}(m) = \pi_2(\delta_i)$,
- $\omega \in S_i$.

Cette propriété étant facilement exprimable en SOM, nous déduisons immédiatement du lemme 4.3.12 :

Lemme 4.3.13. *Soit $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}$ un automate sans-entrée et P-sécable de taille d, avec $\vec{O} \in \mathcal{O}_{\mu_{|P}}$, il existe une SOM-formule* $\mathrm{REG}_{\mathcal{A}}$ *sur $\mu(\mathbf{T}(A)_{|P})^{\vec{O}}$, telle que $\forall S_1,\ldots,S_d \subseteq \mu(P)$,*

1. $\mu(\mathbf{T}(A)_{|P})^{\vec{O}}) \models \mathrm{REG}_{\mathcal{A}}(S_1,\ldots,S_d)$
2. $\vec{S} = (S_1,\ldots,S_d)$ *code une stratégie sans mémoire pour le joueur 0 dans $J_{\mathcal{A}}$.*

Nous avons remarqué précédemment que toute stratégie pour joueur 0 définit un run de l'automate. Nous pouvons également observer que si la stratégie S est positionnelle, alors le run r_{s} décrit par la stratégie est $\vec{S}_{\mathrm{s}}$-régulier. Il suffit pour cela de voir le vecteur $\vec{S}_{\mathrm{s}}$ comme un oracle indiquant pour chaque $r_{\mathrm{s}}(u) = p$, l'unique transition à appliquer (i.e., l'unique δ telle que $\mathrm{S}(\mu(u),p) = (\mu(u),\delta)$).

En transformant le test de chaque transition $\delta_i \in \Delta$ de façon à appliquer une transition δ_i à un noeud u ssi $\mu(u) \in S_i$, nous construisons un automate d'arbre **déterministe** sans entrée $\mathcal{A}_{\mathrm{s}} = (Q,\emptyset,A,{\mu_P}^{-1}(\vec{S}_{\mathrm{s}}),\Delta_{\mathrm{s}},q_0)$, où Δ_{s} est composé de tous les $(q_\perp,\vec{o},q_\perp,q_\perp)$ qui appartiennent à Δ, ainsi que tous les $(p,\vec{b},p_0,p_1)$ tels que $\delta_i = (p,\vec{o},p_0,p_1)$ et $\vec{b} \in \{0,1\}^d$ vérifie $b_i = 1$ et pour tout $j \neq i \in [1,d]$ tel que δ_j commence par p, $b_j = 0$.

L'unique run cet automate (si il existe) est un run acceptant de $\mathcal{A}$ qui est ${\mu_{|P}}^{-1}(\vec{S}_{\mathrm{s}})$-régulier.

Lemme 4.3.14. *Pour toute stratégie sans mémoire* s *gagnante pour le joueur 0, l'automate* $\mathcal{A}_{\mathrm{s}}$ *est déterministe, P-sécable et son unique run est un run acceptant de* $\mathcal{A}$.

Preuve : Clairement, $\mathcal{A}_{\mathrm{s}}$ est déterministe et admet un run puisque pour tout $u \in P$, il existe $i \in [1, d]$ tel que $\pi_1(\delta_i) = r(u)$ et $\mu(u) \in S_{\mathrm{s}_i}$ par définition de du codage de la stratégie. Montrons que cet unique run r est un run acceptant de $\mathcal{A}$:

- $r(\varepsilon) = q_0$
- $\forall u \in A^*$, la transition $(r(u), \chi_{{\mu_{|P}}^{-1}(\vec{S}_{\mathrm{s}})}(u), r(ua_0), r(ua_1))$ appartient à Δ_{s} et il existe donc $i \in [1, \ldots, d]$ et $\vec{o}$ tels que $\delta_i = (r(u), \vec{o}, r(ua_0), r(ua_0)) \in \Delta$ et $\mathsf{S}_i = 1$. Nous avons donc de plus $\mathrm{s}(\mu(u), r(u)) = (\mu(u), \delta_i)$ et donc $\vec{o} = \chi_{\vec{O}}(u)$.

Donc, r est bien un run de $\mathcal{A}$, et comme la stratégie s est gagnante, r est un run acceptant de $\mathcal{A}$ et $\mathcal{A}_{\mathrm{s}}$ est P-sécable.

□

De plus, $\mathcal{A}_{\mathrm{s}}$ étant déterministe, la proposition 4.2.6 implique que son unique run est $\vec{S}_{\mathrm{s}}$-régulier.

En appliquant la proposition 4.2.5, il est possible d'étendre ces résultats au cas général des automates avec entrée. De plus, si P est SOM-définissable dans $\mathbf{T}(A)^{\vec{O}}$ alors d'après le lemme 4.1.24, pour tout automate $\mathcal{A}$ dans $\mathsf{AFA}^{\vec{O}}$, il existe un automate $\mathcal{B} \in \mathsf{AFA}^{\vec{O}}$, P-sécable, tel que $\mathrm{F}(\mathcal{A})_{|P} = \mathrm{F}(\mathcal{A})_{|P}$. Ainsi, les deux résultats précédents peuvent être étendus à tout $\mathcal{A} \in \mathsf{AFA}^{\vec{O}}$ pour $\vec{O} \in \mathcal{O}_{\mu_{|P}}$ tel que P est SOM-définissable dans $\mathbf{T}(A)^{\vec{O}}$.

La proposition suivante résume ces résultats.

Proposition 4.3.15. *Pour toute forêt $F \in \mathsf{TREC}^{\vec{O}}(A)_{|P}$, où $\vec{O} \in \mathcal{O}_{\mu_{|P}}$, et P est SOM-définissable dans $\mathbf{T}(A)^{\vec{O}}$, il existe $d \geq 0$ et une formule REG_F à d variables libres telle que*

1. *$F \neq \emptyset$ ssi $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}}) \models \exists X_1, \ldots X_d.\ \mathrm{REG}_F(X_1, \ldots, X_d)$*
2. *$\forall S_1, \ldots, S_d \subseteq \mu(P)$,*
   si $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}}) \models \mathrm{REG}_F(S_1, \ldots, S_d)$ alors F contient un arbre $\mu_{|P}{}^{-1}(\vec{S})$-régulier.

4.3.3 Théorèmes de transferts

Nous utilisons maintenant la proposition 4.3.15 pour transférer des propriétés de la structure $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ vers la structure $\mathbf{T}(A)_{|P}{}^{\vec{O}}$. Fixons tout d'abord les hypothèses pour lesquelles ces théorèmes sont établis :

Définition 4.3.16 (Hypothèse (HT)). *Soient*

- *$\mu : A^* \to M$ un morphisme de semi-groupe surjectif,*
- *$\vec{R}$ un vecteur de sous-ensembles de M,*
- *$P \subseteq A^*$ un langage clos par préfixe et* SOM*-définissable dans $\mathbf{T}(A)^{{\mu_{|P}}^{-1}(\vec{R})}$.*

Dans ce cas, nous notons $\mathrm{HT}(\mu_{|P}, \vec{O})$*, où $\vec{O} = {\mu_{|P}}^{-1}(\vec{R})$.*

Théorème 4.3.17 (Transfert de décidabilité). *Soient μ un morphisme de A^* vers un semi-groupe quelconque, $P \subseteq A^*$, et $\vec{O}$ un vecteur de sous-ensembles de P tels que* $\mathrm{HT}(\mu_P, \vec{O})$.

Si la SOM*-théorie de $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ est décidable, alors*

- *la* SOM*-théorie de $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ est décidable,*
- *le problème du vide est décidable pour les forêts $\vec{O}$-reconnaissables.*

Preuve : Soit F une forêt $\vec{O}$-reconnaissable et REG_F la formule établie dans la proposition 4.3.15. Si on sait décider si $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}}) \models \exists \vec{X}.\ REG_F(\vec{X})$, on sait décider si F est vide. Alors, en utilisant l'équivalence entre le problème du vide pour les forêt $\vec{O}$-reconnaissables et le problème de la satisfiabilité des SOM-formules closes sur $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ (corollaire 4.1.19), le théorème est démontré.

□

Définissons maintenant la condition permettant le transfert de la propriété MD.

Définition 4.3.18. *Étant donné μ un morphisme de A^* dans un monoïde, et $P \subseteq A^*$ clos par préfixe, l'application restreinte $\mu_{|P}$ est dite* **SOM-inversible** *si $\forall \vec{O} = (O_1, \ldots, O_m)$, $O_i \subseteq P$, $\forall D \subseteq \mu(P)$:*

D SOM-définissable dans $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ implique $\mu_{|P}{}^{-1}(D)$ SOM-définissable dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$.

Théorème 4.3.19 (Transfert de la propriété MD). *Soient μ un morphisme de A^* vers un monoïde quelconque, $P \subseteq A^*$, et $\vec{O}$ un vecteur de sous-ensembles de P tels que* $\text{HT}(\mu_P, \vec{O})$ *et $\mu_{|P}$ est* SOM-*inversible, alors*

$\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ vérifie MD *implique $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ vérifie* MD.

Preuve : D'après le théorème 4.1.21 et puisque $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ vérifie MD, pour toute forêt $\vec{O}$-reconnaissable non vide $F \subseteq P\text{-}\mathsf{Arbre}(\{0,1\})$, il existe $\vec{S}$ SOM-définissable dans $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ tel que F contient un arbre $\mu_{|P}{}^{-1}(\vec{S})$-régulier. Puisque $\mu_{|P}$ est SOM-inversible, $\mu_{|P}{}^{-1}(\vec{S})$ est SOM-définissable dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ et d'après la proposition 4.2.3, $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ vérifie MD.

□

Théorème 4.3.20 (Théorème de structure). *Soient μ un morphisme de A^* vers un monoïde quelconque, $P \subseteq A^*$, et $\vec{O}$ un vecteur de sous-ensembles de P tels que* $\text{HT}(\mu_P, \vec{O})$ *et $\mu_{|P}$ est* SOM-*inversible, et $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ vérifie* MD, *alors pour tout $L \subseteq P$,*

L est SOM-définissable dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ ssi

$\exists \vec{D}$ SOM-définissable dans $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ tel que L est $\mu_{|P}{}^{-1}(\vec{D})$-régulier.

Preuve : Supposons que L est défini dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ par une formule SOM $\phi(X)$. Cette formule définit dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$ la forêt $F = \{\text{T}(A)^L{}_{|P}\}$ et d'après le théorème 4.1.21, F est $\vec{O}$-reconnaissable. Puisque par l'hypothèse, $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$ vérifie MD, la proposition 4.2.3 assure de l'existence d'un vecteur $\vec{D}$, SOM-définissable dans $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$, et tel que l'arbre $\text{T}(A)^L{}_{|P}$ est $\mu_{|P}{}^{-1}(\vec{D})$-régulier. Par définition d'arbre régulier, le langage L est $\mu_{|P}{}^{-1}(\vec{D})$-régulier.

Inversement, si $\vec{D}$ est SOM-définissable dans $\mu(\mathbf{T}(A)_{|P}{}^{\vec{O}})$, alors puisque $\mu_{|P}$ est SOM-inversible, $\mu_{|P}{}^{-1}(\vec{D})$ est SOM-définissable dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$. Étant donné $L \in \mathsf{REC}^{\mu_{|P}{}^{-1}(\vec{D})}$, en utilisant la caractérisation par automate de L, il est alors facile de trouver une formule SOM définissant L dans $\mathbf{T}(A)_{|P}{}^{\vec{O}}$.

□

Remarque 4.3.21. *Dans le théorème 4.3.20, il n'est pas nécessaire de supposer que μ_P est SOM-inversible pour prouver que les définissables sont reconnus par automates à oracles. Cette hypothèse n'est nécessaire que pour l'implication inverse.*

4.3.4 Un exemple d'application

Différents auteurs ont exhibés des classes de prédicats $R \subseteq \mathbb{N}$ pour lesquels la structure $\langle \mathbb{N}, 0, +1, R\rangle$ admet une SOM théorie décidable. Par exemple, la classe des prédicats morphiques étudiée dans [CT02], ou celle étudiée dans le chapitre 13 de ce manuscrit. Ces structures peuvent être vues comme des structures images par un morphisme, ce qui permet transférer la propriété de décidabilité vers des structures d'arbres. Étant donnés deux alphabets A et B, avec $B \subseteq A$, l'application $l_B : A^* \to \mathbb{N}$ qui à tout mot associe son nombre de lettre dans B est un morphisme surjectif.

Corollaire 4.3.22. *Pour tout $\vec{N} = (N_1, \ldots, N_m)$, $N_i \subseteq \mathbb{N}$ tel que $\langle \mathbb{N}, +1, \vec{N}\rangle$ admet une* SOM*-théorie décidable, la structure $\langle A^*, (\mathrm{SUCC}_a)_{a\in A}, {l_B}^{-1}(\vec{N})\rangle$ admet une* SOM*-théorie décidable.*

Ce résultat a déjà été prouvé dans [Tho03, proposition 2], pour le cas $B = A$, comme une application directe des résultats sur les *dépliages* de graphes obtenus dans [Cou95, CW98] : si un graphe admet une théorie SOM décidable, alors l'arbre obtenu par dépliage du graphe également. La structure $\langle A^*, (\mathrm{SUCC}_a)_{a\in A}, {l_A}^{-1}(R)\rangle$ peut être vue comme le dépliage du graphe $\langle \mathbb{N}, +1_{a_1}, \ldots, +1_{a_n}, R\rangle$, où $A = \{a_1, \ldots, a_n\}$ et les relations $+1_{a_i}$ sont toutes égales à la relation $+1$. Toutefois, à notre connaissance, il n'est pas prouvé que l'opération de dépliage préserve la propriété MD.

Corollaire 4.3.23. *Pour tout vecteur $\vec{N}$ de sous-ensembles de $\mathbb{N}$ tel que $\langle \mathbb{N}, 0, +1, \vec{N}\rangle$ vérifie MD,*

1. *la structure $\langle A^*, \varepsilon, (\mathrm{SUCC}_a)_{a\in A}, {l_A}^{-1}(\vec{N})\rangle$ vérifie MD,*
2. *les ensembles définissables dans $\langle A^*, \varepsilon, (\mathrm{SUCC}_a)_{a\in A}, {l_A}^{-1}(\vec{N})\rangle$ coïncident exactement avec les langages reconnus par automates dans $\mathsf{AF}^{{l_A}^{-1}(\vec{D})}$, pour $\vec{D}$ définissable dans $\langle \mathbb{N}, 0, +1, \vec{N}\rangle$.*

Preuve : Ces résultats sont immédiats en appliquant les théorème 4.3.19 et 4.3.20. Pour pouvoir les appliquer, il suffit de vérifier que le morphisme l est SOM-inversible. Nous commençons pour cela par définir un recouvrement de A^* tel que, l est une bijection sur chaque élément du recouvrement, et la propriété "être un élément du recouvrement" est exprimable en LSOM dans la structure $\mathbf{T}(A)$. Nous choisissons $\mathcal{R}$ formé de tous les R constituant un chemin infini issu de ε, c'est à dire qu'il existe un mot infini w tel que R est composé exactement de tous les préfixes finis de w. Ce recouvrement vérifie clairement les propriétés annoncées.

Pour toute formule ϕ à n variable libres sur $\mathcal{S} = \langle \mathbb{N}, 0, +1, R\rangle$, nous construisons par induction sur la structure de ϕ, une formule ϕ' sur $\mathbf{T}(A)^{l^{-1}\vec{N}}$ à $n+1$ variable libre vérifiant pour tout $R \in \mathcal{R}$, pour tous $S_1, \ldots, S_n \subseteq A^*$:

$$\mathbf{T}(A)^{l^{-1}(\vec{N})} \models \phi'(S_1 \cap R, \ldots, S_n \cap R, R) \text{ ssi } \mathcal{S} \models \phi(l(S_1 \cap R, \ldots, S_n \cap R)). \tag{1}$$

- $0'(x, Y) := \varepsilon(x)$,
- $(+1)'(x_1, x_2, Y) := \bigvee_{a\in A} \mathrm{SUCC}_a(x_1, x_2)$,
- si $\varphi := \exists X, \psi(X, X_1, \ldots, X_n)$, alors $\varphi' := \exists X, \psi'(X \cap Y, X_1, \ldots, X_n, Y)$,
- si $\varphi := \forall X, \psi(X, X_1, \ldots, X_n)$, alors $\varphi' := \exists X, \psi'(X \cap Y, X_1, \ldots, X_n, Y)$,

les cas des formules $X_1 \subseteq X_2$ et $X_1 \subseteq N$ sont donnés de façon évidente, les cas de combinaisons booléennes également : $(\phi \vee \psi)' := \phi' \vee \psi'$, $(\phi \wedge \psi)' := \phi' \wedge \psi'$ et $(\neg\phi)' := \neg\phi'$. Il est assez simple de vérifier que les formules ainsi construites satisfont bien l'équivalence (1). Nous traitons ici simplement le cas de la quantification existentielle, celui de la quantification universelle est identique.

Soient $S_1, \ldots, S_n \subseteq A^*$ et $R \in \mathcal{R}$. Par construction,

$$\begin{array}{rl}
\mathbf{T}(A)^{l^{-1}(\vec{N})} \models \phi'(S_1 \cap R, \ldots, S_n \cap R, R) & \text{ssi} \\
\exists S \subseteq A^*,\ \mathbf{T}(A)^{l^{-1}(\vec{N})} \models \psi'(S \cap R, S_1 \cap R, \ldots, S_n \cap R, R) & \text{ssi (par h.i.)} \\
\exists S \subseteq A^*,\ \mathcal{S} \models \psi(l(S \cap R), l(S_1 \cap R), \ldots, l(S_n \cap R)) & \text{ssi (car } l_{|R} \text{ est bijective)} \\
\exists D \subseteq \mathbb{N},\ \mathcal{S} \models \psi(D, l(S_1 \cap R), \ldots, l(S_n \cap R)) & \text{ssi} \\
\mathcal{S} \models \exists X,\ \psi(X, l(S_1 \cap R), \ldots, l(S_n \cap R)). &
\end{array}$$

Nous avons remarqué que $\mathcal{F}_{k+1}$ forme un recouvrement de A^* et que la restriction de l à tout $R \in \mathcal{R}$ est bijective. Donc $\forall D \subseteq A^*$, $D' \subseteq \mathbb{N}$,

$$D = l^{-1}(D') \text{ ssi } \forall R \in \mathcal{R},\ l(R \cap D) = D'.$$

Alors, pour toute formule $\phi(X_1, \ldots, X_n)$ sur $\mathcal{S}$, la formule

$$\phi_l(X_1, \ldots, X_n) := \forall R \in \mathcal{R},\ \phi'(R \cap X_1, \ldots, R \cap X_n, R)$$

satisfait $\mathbf{T}(A)^{l^{-1}(\vec{N})} \models \phi_l(D_1, \ldots, D_n)$ ssi $\exists D'_1, \ldots, D'_n$ tels que $\mathcal{S} \models \phi(D'_1, \ldots, D'_n)$ et pour tout $i \in [1, n]$, $l^{-1}(D_i) = D'_i$. La proposition est donc prouvée.

□

Suite à de récents travaux non encore publiés de A. Rabinovich [Rab05], il semble que pour tout R, la structure $\langle \mathbb{N}, 0, +1, R \rangle$ vérifie la propriété MD (qu'il nomme propriété de "sélection"). Les conséquences du corollaire ci-dessus seraient donc vraies pour tout prédicat R.

Chapitre 5

Ensembles de mots k-réguliers

Les résultats obtenus dans le chapitre précédent (théorèmes 4.3.17, 4.3.19 et 4.3.20) ont été établis pour des structures ayant pour domaine un ensemble de mots. Nous ne pouvons donc pas les appliquer directement sur les ensembles de piles itérées. Toutefois, nous avons mis en évidence dans le chapitre §1.4 le fait que l'ensemble k-$\mathsf{Pile}(A_1, \ldots, A_k)$ est en bijection avec un ensemble de mots **clos par préfixe** noté $\mathcal{P}_k(A_1, \ldots, A_k)$. Nous définissons la structure $\boldsymbol{\mathcal{P}_k}(A_1, \ldots, A_k)$ dont le domaine est $\mathcal{P}_k(A_1, \ldots, A_k)$, et les relations sont celles induites par le produit à gauche par une lettre dans le groupe libre $\mathrm{Irr}(A_{1,k})$. (Rappelons que $A_{1,k}$ désigne l'union de $A_1, \ldots, A_k$ et que les définitions de base sur le groupe libre sont introduites dans §0.1.)

Nous prouvons que la structure $\boldsymbol{\mathcal{P}_k}(A_1, \ldots, A_k)$ admet une SOM-théorie décidable, vérifie la propriété MD et définissons une classe d'automates à oracles reconnaissant exactement les définissables de $\boldsymbol{\mathcal{P}_k}(A_1, \ldots, A_k)$.

Nous utilisons pour cela l'application $\mathrm{f}_k : \mathcal{P}_{k+1} \to \mathcal{P}_k$ (définition 1.4.1) qui correspond à la restriction à $\mathcal{P}_{k+1}$ d'un morphisme surjectif des mots sur l'alphabet $\widehat{A_{1,k+1}}$ dans le groupe $(\mathrm{Irr}(A_{1,k}), \bullet, \varepsilon)$.

Les théorèmes de transfert s'appliquent de la structure $\mathrm{f}_k(\boldsymbol{\mathcal{P}_{k+1}})$ vers la structure d'arbre de domaine $\mathcal{P}_{k+1}$. Les structures $\mathrm{f}_k(\boldsymbol{\mathcal{P}_{k+1}})$ et $\boldsymbol{\mathcal{P}_k}$ et $\mathbf{T}(\widehat{A_{1,k}})_{|\mathcal{P}_k}$ étant équivalentes pour la logique SOM, les théorèmes peuvent être utilisés itérativement pour tout $k \geq 1$.

Une condition nécessaire à l'application des théorèmes 4.3.19 et 4.3.20 est que f_k doit être SOM-inversible (voir définition 4.3.18), nous prouvons que cette condition est vérifiée dans la première section, nous appliquons ensuite les théorèmes de transferts dans la seconde section.

5.1 Logique dans un groupe libre

Cette section est dédiée à la preuve de la proposition 5.1.4 établissant que l'image inverse par f_k d'un ensemble SOM-définissable dans $\boldsymbol{\mathcal{P}_k}$, est SOM-définissable dans $\boldsymbol{\mathcal{P}_{k+1}}$.

Soit $\boldsymbol{\mathcal{P}_k}(A_1, \ldots, A_k)$ (ou $\boldsymbol{\mathcal{P}_k}$) la structure sur Sig_k définie de la façon suivante :

$$\boldsymbol{\mathcal{P}_k}(A_1, \ldots, A_k) = \langle \mathcal{P}_k(A_1, \ldots, A_k),\ \varepsilon,\ (\bullet_a)_{a \in \widehat{\mathcal{A}_k}} \rangle$$

où $\forall a \in \widehat{\mathcal{A}_k}$, $\bullet_a = \{(u, v) |\ u, v \in \mathcal{P}_k,\ v = u \bullet a\}$.
Nous procédons de façon similaire à la preuve du corollaire 4.3.23. Nous commençons par

recouvrir l'ensemble $\mathcal{P}_{k+1}$ par une famille d'ensembles en bijection avec $\mathcal{P}_k$. Définissons pour tout $k \geq 1$, la famille $\mathcal{F}_{k+1}$ de langages dans $\mathcal{P}_{k+1}$ vérifiant la propriété suivante : $\forall F \subseteq \mathcal{P}_{k+1}$, $F \in \mathcal{F}_{k+1}$ ssi

- soit $F = \mathcal{P}_k$
- soit $\exists u \in \mathcal{P}_{k+1}$ et $a \in A_{k+1}$ tels que $F = \{uaw \in \mathcal{P}_{k+1} |\ w \in \mathrm{Irr}(A_{1,k})\}$.

Ce découpage de $\mathcal{P}_{k+1}$ va nous permettre de reconstituer l'image inverse d'un sous-ensemble définissable de $\mathcal{P}_k$. La preuve repose sur les remarques suivantes :

Remarque 5.1.1. *Pour tout $k \geq 1$:*

1. *$\mathcal{F}_{k+1}$ forme une partition de $\mathcal{P}_{k+1}$,*
2. *Pour tout $F \in \mathcal{F}_{k+1}$, la restriction de f_k à F est une bijection.*
   Si $F = \mathcal{P}_k$, c'est évident. Sinon, F s'écrit $F = \{uaw \in \mathcal{P}_{k+1} |\ w \in \mathrm{Irr}(A_k)\}$, et pour tout $v \in \mathcal{P}_k$, qu'il existe un et un seul $w \in \mathrm{Irr}(A_{1,k})$ tel que $\mathrm{f}_k(uaw) = v$. En effet, plongeons $\mathcal{P}_k$ dans le groupe libre $\mathrm{Irr}(A_{1,k})$: il existe un et un seul $w' \in \mathrm{Irr}(A_{1,k})$ tel que $\mathrm{f}_k(ua) \bullet w' = v$. Nous savons calculer la valeur de ce mot : $w' = \overline{\mathrm{f}_k(ua)} \bullet v$.
   Puisque pour tout $w \in \mathrm{Irr}(A_{1,k})$, $\mathrm{f}_k(uaw) = \mathrm{f}_k(ua) \bullet w$, le mot $w' = \overline{f_k(ua)} \bullet v$ est l'unique élément de $\mathrm{Irr}(A_{1,k})$ tel que $\mathrm{f}_k(uaw) = v$.
3. *La propriété $F \in \mathcal{F}_{k+1}$ est SOM-exprimable dans $\boldsymbol{\mathcal{P}_{k+1}}$:*
   en effet, $F \in \mathcal{F}_{k+1}$ ssi F est une partie maximale telle que pour tout $u, v \in F$, il existe un chemin de u à v n'utilisant que des arcs $\bullet_a$ pour $a \in \widehat{A_{1,k}}$. Il est donc possible de construire une SOM-formule dont l'ensemble des modèles dans $\boldsymbol{\mathcal{P}_{k+1}}$ est $\mathcal{F}_{k+1}$.

Lemme 5.1.2. *Pour toute formule à m places $\phi(X_1, \ldots, X_n)$ sur Sig_k, $k \geq 1$, il existe une formule à m places $\phi'(X_1, \ldots, X_n, Y)$ sur Sig_{k+1} telle que $\forall R_1, \ldots, R_m \subseteq \mathcal{P}_k$:*
$\forall S_1, \ldots, S_n \in \mathcal{P}_{k+1}$, $\forall F \in \mathcal{F}_{k+1}$,

$$\boldsymbol{\mathcal{P}_{k+1}}^{\mathrm{f}_k{}^{-1}(R_1,\ldots,R_m)} \models \phi'(S_1 \cap F, \ldots, S_n \cap F, F) \quad ssi \quad \boldsymbol{\mathcal{P}_k}^{(R_1,\ldots,R_m)} \models \phi(\mathrm{f}_k(S_1 \cap F, \ldots, S_1 \cap F))$$

Comme nous en avons pris l'habitude, f_k et $\mathrm{f}_k{}^{-1}$ désignent également leur généralisation aux vecteurs de sous-ensembles :
$\mathrm{f}_k{}^{-1}(R_1, \ldots, R_m) = (\mathrm{f}_k{}^{-1}(R_1), \ldots, \mathrm{f}_k{}^{-1}(R_m))$.
Preuve : Pour toute formule ϕ, la formule ϕ' est construite par induction sur la structure de ϕ. Nous ne donnons ici que la construction pour ϕ formule atomique, les autres cas sont donnés exactement par la même induction que dans la preuve du corollaire 4.3.23.

- $(\varepsilon)'(x, Y) := \bigwedge_{a \in A_{1,k}} \neg(\exists y,\ x = y \bullet a)$,
- $\forall a \in \widehat{A_{1,k}}$, $(\bullet_a)'(x_1, x_2, Y) := \bullet_a(x_1, x_2)$,
- si $\phi(X) := X \subseteq R_i$, alors $\phi'(X, Y) := X \subseteq \mathrm{f}_k^{-1}(R_i)$,
- si $\phi(X_1, X_2) := X_1 \subseteq X_2$, alors $\phi'(X_1, X_2, Y) := X_1 \subseteq X_2$.

Soit $\vec{R} = (R_1, \ldots, R_m)$ avec $R_i \subseteq \mathcal{P}_k$ et $\phi \in \mathrm{SOM}(Sig_k)$, une induction sur la structure de ϕ permet de vérifier que ϕ' satisfait bien le lemme. Nous examonons le cas où ϕ est une formule atomique, les autres cas se vérifient comme dans la preuve du corollaire 4.3.23.

- Une simple induction permet de vérifier que pour tout $u \in \mathcal{P}_{k+1}$ satisfaisant $(\varepsilon)'(x, F)$, $f_k(u) = \varepsilon$: en effet, $f_k(u) = \varepsilon$ ssi $u \bullet \bar{a} \notin \mathcal{P}_k$, $\forall a \in A_k$ et $f_{k-1}(u) = \varepsilon$.
- Soit $a \in \widehat{A_{1,k}}$. Si $F = \mathcal{P}_k$, alors nous avons clairement, pour tous $u, v \in \mathcal{P}_k$

$$\boldsymbol{\mathcal{P}_{k+1}}^{\mathrm{f}_k{}^{-1}(\vec{R})} \models \bullet_a{}'(u, v, F) \text{ ssi } \boldsymbol{\mathcal{P}_k}^{\vec{R}} \models \bullet_a(\mathrm{f}_k(u), \mathrm{f}_k(v)).$$

Si $F = \{ubw \in \mathcal{P}_{k+1} |\ w \in \mathrm{Irr}(A_k)\}$, avec $b \in A_{k+1}$, alors pour tous $v = ubw$, $v' = ubw' \in F$,

$$\begin{array}{lll} \boldsymbol{\mathcal{P}_{k+1}}^{\mathrm{f}_k{}^{-1}(\vec{R})} \models (\bullet_a)'(v, v', F) & \text{ssi} & \boldsymbol{\mathcal{P}_{k+1}}^{\mathrm{f}_k{}^{-1}(\vec{R})} \models \bullet_a(ubw, ubw') \\ & \text{ssi} & w' = w \bullet a \\ & \text{ssi} & \mathrm{f}_k(ub) \bullet w' = \mathrm{f}_k(ub) \bullet w \bullet a \\ & \text{ssi} & \boldsymbol{\mathcal{P}_k}^{\vec{R}} \models \bullet_a(\mathrm{f}_k(v), \mathrm{f}_k(v')). \end{array}$$

– Si $\phi(X) = X \subseteq R_1$, alors pour tout $S \subseteq \mathcal{P}_{k+1}$,

$$\begin{array}{lll} \boldsymbol{\mathcal{P}_{k+1}}^{\mathrm{f}_k{}^{-1}(\vec{R})} \models \phi'(S \cap F, F) & \text{ssi} & \boldsymbol{\mathcal{P}_{k+1}}^{\mathrm{f}_k{}^{-1}(\vec{R})} \models S \cap F \subseteq \mathrm{f}_k{}^{-1}(R_1) \\ & \text{ssi} & \mathrm{f}_k(S \cap F) \subseteq R_1 \\ & \text{ssi} & \boldsymbol{\mathcal{P}_k}^{\vec{R}} \models \phi(\mathrm{f}_k(u), \mathrm{f}_k(v)). \end{array}$$

– Si $\phi(X, Y) = X \subseteq Y$, nou procédons exactement comme pour le cas précédent.

□

Maintenant que nous savons définir des parties de l'image inverse d'un sous-ensemble de $\mathcal{P}_k$, il suffit de toutes les réunir pour reconstituer une image inverse complète.

Proposition 5.1.3. *Pour toute* SOM*-formule à m places $\phi(X_1, \ldots, X_n)$ sur Sig_k, $k \geq 1$, il existe une formule* SOM *à m places $\phi^{+1}(X_1, \ldots, X_n)$ sur Sig_{k+1} telle que*
$\forall R_1, \ldots, R_m \subseteq \mathcal{P}_k$, $\forall S_1, \ldots, S_n \in \mathcal{P}_{k+1}$,

$$\boldsymbol{\mathcal{P}_{k+1}}^{\mathrm{f}_k{}^{-1}(R_1, \ldots, R_m)} \models \phi^{+1}(S_1, \ldots, S_n)$$
$$\textit{ssi}$$
$$\exists D_1, \ldots, D_n \subseteq \mathcal{P}_k \textit{ tels que } \boldsymbol{\mathcal{P}_k}^{(R_1, \ldots, R_m)} \models \phi(D_1, \ldots, D_n) \textit{ et } \forall i,\ S_i = \mathrm{f}_k^{-1}(D_i).$$

Preuve : La construction est identique à celle de la formule ϕ_l donnée dans la preuve du corollaire 4.3.23.

□

Proposition 5.1.4. *Pour tout $k \geq 1$, pour tout $\vec{R}$ vecteur de sous-ensembles de $\mathcal{P}_k$, pour tout $D \subseteq \mathcal{P}_k$ définissable dans $\boldsymbol{\mathcal{P}_k}^{\vec{R}}$ l'ensemble $\mathrm{f}_k^{-1}(D)$ est définissable dans $\boldsymbol{\mathcal{P}_{k+1}}^{\mathrm{f}_k{}^{-1}(\vec{R})}$.*

5.2 Langages k-réguliers

Nous appliquons maintenant les théorèmes de transfert à la structure $\boldsymbol{\mathcal{P}_k}$. Nous prouvons qu'elle admet une SOM-théorie décidable et vérifie la propriété MD. Nous en déduisons une caractérisation des langages SOM-définissables dans $\boldsymbol{\mathcal{P}_k}$ par automate finis à oracle. Nous nommons ces langages : **langages k-réguliers**.

Toutes les notions relatives au langage $\mathcal{P}_k$ sont définies au §1.4.
Commençons par remarquer que l'application $\mathrm{f}_k : \mathcal{P}_{k+1} \to \mathcal{P}_k$ est la restriction du morphisme $\mu_k = \rho \circ \pi_{\widehat{A_{1,k}}} : \widehat{A_{1,k+1}}^* \to \mathrm{Irr}(A_{1,k})$ à l'ensemble $\mathcal{P}_{k+1}$. Nous introduisons pour tout $k \geq 1$, le vecteur $\vec{O_k}$ de sous-ensembles de $\mathcal{P}_k$ défini inductivement par :

– $\vec{O_1} = \emptyset$
– $\vec{O_{k+1}} = (\mathrm{f}_k{}^{-1}(\vec{O_k}), \mathrm{f}_k{}^{-1}(\mathcal{P}_k \cdot a_1), \ldots, \mathrm{f}_k{}^{-1}(\mathcal{P}_k \cdot a_n))$, où $A_{1,k} = \{a_1, \ldots, a_n\}$

En d'autres mots, $\vec{O}_{k+1}$ est composé de tous les $\mathrm{fin}_{k,i}{}^{-1}(a)$, pour $1 \leq i \leq k$ et $a \in \mathcal{A}_i$. Notons $\boldsymbol{S_k} = \langle \mathcal{P}_k, \varepsilon, (\mathrm{SUCC}_a)_{a \in \widehat{A_k}} \rangle$.

Lemme 5.2.1. *Pour tout $k \geq 1$, les structures $\boldsymbol{\mathcal{P}_k}$, $\boldsymbol{S_k}$, $\boldsymbol{S_k}^{\vec{O}_k}$ et $\mathrm{f}_k(\boldsymbol{S_{k+1}}^{\vec{O}_{k+1}})$ sont* SOM*-équivalentes.*

Preuve :

- Clairement, $\boldsymbol{\mathcal{P}_k}$ est définissable dans $\boldsymbol{S_k}$ puisque $\forall u, v \in \mathcal{P}_k$, $a \in \widehat{A_{1,k}}$, $u = v \bullet a$ ssi $u = va$ ou $v = u\bar{a}$. Inversement, pour montrer que $\boldsymbol{S_k}$ est définissable dans $\boldsymbol{\mathcal{P}_k}$, nous prouvons tout d'abord que si $\widehat{A_{1,k}} = \{a_1, \ldots, a_n\}$, alors le vecteur $(\mathcal{P}_k \cdot a_1, \ldots, \mathcal{P}_k \cdot a_n)$ est SOM-définissable dans $\boldsymbol{\mathcal{P}_k}$.
  Il suffit pour cela de remarquer que chaque $\mathcal{P}_k \cdot a_i$ est le plus petit sous-ensemble S_i de $\mathcal{P}_k$ vérifiant pour tout $u \in \mathcal{P}_k$:
  $u \in S_i$ ssi
  - soit $\bullet_{a_i}(u, \varepsilon)$,
  - soit il existe $a_j \neq \overline{a_i} \in \widehat{A_k}$ et $v \in S_j$ tels que $u = v \bullet a_i$.

  Il est alors simple de définir dans $\boldsymbol{\mathcal{P}_k}$ la relation induite par la concaténation puisque pour tous $u, v \in \mathcal{P}_k$, $a \in \widehat{A_k}$, $[u = va$ ssi $u = v \bullet a$ et $v \notin \mathcal{P}_k \cdot \overline{a}]$.
- $\boldsymbol{S_k}^{\vec{O}_k}$ est définissable dans $\boldsymbol{\mathcal{P}_k}$ puisque d'après le point précédent, $\forall 1 \leq i \leq k$, $a \in A_{1,i}$, l'ensemble $\mathcal{P}_i \cdot a$ est SOM-définissable dans $\boldsymbol{\mathcal{P}_i}$ et donc d'après la proposition 5.1.4, $\mathrm{f}_{k,i}{}^{-1}(\mathcal{P}_i \cdot a)$ est SOM-définissable dans $\boldsymbol{\mathcal{P}_k}$.
- $\mathrm{f}_k(\boldsymbol{S_{k+1}}^{\vec{O}_{k+1}}) = \langle \mathcal{P}_k, \varepsilon, (\bullet_{\pi_{\widehat{A_{1,k}}}(a)})_{a \in \widehat{A_{1,k+1}}}, \vec{O}_k \rangle$. Cette structure est équivalente à la structure $\boldsymbol{\mathcal{P}_k}^{\vec{O}_k}$, qui est elle même équivalente à la structure $\boldsymbol{\mathcal{P}_k}$.

□

Nous utiliserons donc indifféremment chacune de ces quatre structures. Vérifions maintenant que pour tout k, le vecteur $O_{k+1}^{\rightarrow}$ et l'application f_k satisfont la propriété $\mathrm{HT}(f_k, O_{k+1}^{\rightarrow})$, nécessaire à l'application des théorèmes de transferts (définition 4.3.16).

Lemme 5.2.2. *Pour tout $k \geq 1$, la condition $\mathrm{HT}(f_k, O_{k+1}^{\rightarrow})$ est satisfaite.*

Preuve : Il suffit de vérifier que pour tout $k \geq 0$, l'ensemble $\mathcal{P}_{k+1}$ est SOM-définissable dans la structure $\mathbf{T}(\widehat{A_{1,k+1}})^{O_{k+1}^{\rightarrow}}$. D'après la proposition 1.4.4, pour tout $u \in \widehat{A_{1,k+1}}^*$, $u \in \mathcal{P}_{k+1}$ ssi un des cas suivants est vérifié :

- $u = \varepsilon$,
- il existe $v \in \mathcal{P}_{k+1}$ et $a \in A_{k+1}$ tels que $u = v \cdot a$,
- il existe $v \in \mathcal{P}_{k+1}$ et $a \in A_{1,k}$ tels que $u = v \cdot a$ et v ne termine pas par $\bar{a}$,
- il existe $v \in \mathcal{P}_{k+1}$ et $a \in A_i$, $i \in [1, k]$ tels que $u = v \cdot \bar{a}$, v ne termine pas par a et appartient à $\mathrm{f}_{k,i}{}^{-1}(\mathcal{P}_i \cdot a)$.

Chaque $\mathrm{f}_{k+1,i}{}^{-1}(\mathcal{P}_i \cdot a)$ faisant partie des composantes de $O_{k+1}^{\rightarrow}$, chacune de ces propriété est exprimable par une formule SOM, et l'ensemble $\mathcal{P}_k$ est donc définissable dans $\mathbf{T}(\widehat{A_{1,k+1}})^{O_{k+1}^{\rightarrow}}$.

□

Théorème 5.2.3. *Pour tout $k \geq 1$, la structure $\boldsymbol{\mathcal{P}_k}$ admet une* SOM*-théorie décidable et vérifie la propriété du Modèle Définissable.*

Preuve : Prouvons ce résultat par induction sur $k \geq 1$.
Base : d'après le théorème 4.2.4, $\boldsymbol{\mathcal{P}_1}$ admet une SOM-théorie décidable et vérifie la propriété MD.
Pas d'induction : supposons que $\boldsymbol{\mathcal{P}_k}$ admet une SOM-théorie décidable et vérifie MD, pour $k \geq 1$. Alors, puisque d'après le lemme 5.2.2, la propriété $\mathrm{HT}(f_k, \vec{O}_{k+1})$ est satisfaite, en utilisant le théorème 4.3.17, et les différentes SOM-équivalences entre les structures, la SOM-théorie de $\boldsymbol{\mathcal{P}_{k+1}}$ est décidable. De la même façon, puisque d'après la proposition 5.1.4, l'application f_k est SOM-inversible, le théorème 4.3.19 implique que $\boldsymbol{\mathcal{P}_{k+1}}$ satisfait la propriété MD.
□

Le même type de raisonnement peut être appliqué à la structure $\boldsymbol{\mathcal{P}_k}^{\vec{O}}$ pour tout vecteur $\vec{O}$ de sous-ensembles de $\mathcal{P}_k$: si la théorie SOM de $\boldsymbol{\mathcal{P}_k}^{\vec{O}}$ est décidable, alors celle de $\boldsymbol{\mathcal{P}_{k+1}}^{\mathrm{f}_k{}^{-1}(\vec{O})}$ aussi.

Théorème 5.2.4. *Soit $\vec{R}$ un vecteur de sous-ensembles de A_1^*,*

1. *si la théorie* SOM *de $\langle {A_1}^*, \varepsilon, (\mathrm{SUCC}_a)_{a\in A_1}, \vec{R}\rangle$ est décidable, alors pour tout $k \geq 1$, la théorie* SOM *de $\boldsymbol{\mathcal{P}_k}^{\mathrm{f}_{k,1}{}^{-1}(\vec{R})}$ est décidable,*
2. *si $\langle A_1^*, \varepsilon, (\mathrm{SUCC}_a)_{a\in A_1}, \vec{R}\rangle$ vérifie la propriété* MD*, alors $\boldsymbol{\mathcal{P}_k}^{\mathrm{f}_{k,1}{}^{-1}(\vec{R})}$ vérifie également la propriété* MD*.*

Définissons maintenant la classe AF_k des automates reconnaissant les langages SOM-définissables dans $\boldsymbol{\mathcal{P}_k}$ ainsi que la classe REC_k des langages reconnus par ces automates. Un automate dans AF_{k+1} est un automate fini dont le vecteur d'oracles est composé d'ensembles de la forme $\mathrm{f}_k{}^{-1}(R)$ où R est lui-même reconnu par un automate dans AF_k.

Définition 5.2.5. *Pour toute suite $(A_\ell)_{\ell\geq 1}$ alphabets disjoints, pour tout $k \geq 0$, les classes $\mathsf{AF}_k(A_1, \ldots, A_k)$ et $\mathsf{REC}_k(A_1, \ldots, A_k)$ sont définies inductivement par :*

- $\mathsf{REC}_0 = \{\{\varepsilon\}, \emptyset\}$
- *pour tout $k \geq 0$,*
  $\mathsf{AF}_{k+1}(A_1, \ldots, A_{k+1})$ *est l'union des classes $\mathsf{AF}^{\mathrm{f}_k{}^{-1}(\vec{R})}(A_1, \ldots, A_{k+1})$, pour $\vec{R}$ vecteur d'éléments de $\mathsf{REC}_k(A_1, \ldots, A_k)$,*
  $\mathsf{REC}_{k+1}(A_1, \ldots, A_k)$ *est l'ensemble des langages dans $\mathcal{P}_{k+1}(A_1, \ldots, A_{k+1}$ reconnus par automates dans $\mathsf{AF}_{k+1}(A_1, \ldots, A_k)$.*

Remarque 5.2.6. *Une définition équivalente pour REC_k est : REC_k est l'ensemble des $\mathrm{L}(\mathcal{A}) \cap \mathcal{P}_k$ tels que $\mathcal{A} \in \mathsf{AF}_k$.*

Théorème 5.2.7. *Pour tout $L \in \widehat{A_k}^*$, $k \geq 1$*
L est définissable en LSOM *dans $\boldsymbol{\mathcal{P}_k}$ ssi $L \in \mathsf{REC}_k$.*

Preuve : Prouvons ce résultat par induction sur $k \geq 1$.
Base : le cas $k = 1$ est évident.
Pas d'induction : supposons le théorème valide pour $k \geq 1$. Le théorème 5.2.4 assure que la structure $\boldsymbol{\mathcal{P}_k}$ vérifie la propriété MD.

Étant donné $D \subseteq \mathcal{P}_{(k+1)}$, le théorème 4.3.20 s'applique, puisque f_k est SOM-inversible, et donc tout ensemble D est SOM-définissable dans $\boldsymbol{\mathcal{P}_k}$ ssi il existe un vecteur $\vec{L}$ d'ensembles SOM-définissables dans $\boldsymbol{\mathcal{P}_k}$ tel que $D \in \mathsf{REC}^{\vec{L}}$. En appliquant l'hypothèse d'induction à chaque L_i, nous obtenons que D est SOM-définissable dans $\boldsymbol{\mathcal{P}_{k+1}}$ ssi $D \in \mathsf{REC}_{k+1}$.
□

Nous étendons ce résultats en permettant aux automates de contenir des oracles construits itérativement à partir du niveau 1.

Définition 5.2.8. *Pour toute suite $(A_\ell)_{\ell\geq 1}$ alphabets disjoints, pour tout $k \geq 1$, pour tout vecteur $\vec{N}$ d'ensembles d'entiers naturels, les classes $\mathsf{AF}_k^{\vec{N}}(A_1,\ldots,A_k)$ et $\mathsf{REC}_k^{\vec{N}}(A_1,\ldots,A_k)$ sont définies inductivement par :*

- $\mathsf{REC}_0^{\vec{N}} = \{\{\varepsilon\},\emptyset\}$,
- *pour tout $k \geq 0$,*

  $\mathsf{AF}_{k+1}^{\vec{N}}(A_1,\ldots,A_{k+1})$ *est l'union des classes* $\mathsf{AF}^{\mathrm{f}_k{}^{-1}(\vec{R})}(A_1,\ldots,A_{k+1})$, *pour $\vec{R}$ vecteur d'éléments de* $\mathsf{REC}_k^{\vec{N}}(A_1,\ldots,A_k)$,

  $\mathsf{REC}_{k+1}^{\vec{N}}(A_1,\ldots,A_k)$ *est l'ensemble des langages dans* $\mathcal{P}_{k+1}(A_1,\ldots,A_{k+1}$ *reconnus par automates dans* $\mathsf{AF}_{k+1}^{\vec{N}}(A_1,\ldots,A_k)$.

Le théorème 5.2.7 s'étend à celle nouvelle que la condition que $\langle \mathbb{N}, +1, \vec{N}\rangle$ vérifie la propriété MD.

Théorème 5.2.9. *Si la structure $\langle \mathbb{N}, +1, \vec{N}\rangle$ vérifie la propriété* MD, *alors pour tout $k \geq 1$, pour tout $L \in \widehat{A_k}^*$, L est* SOM*-définissable dans $\boldsymbol{\mathcal{P}}_{\mathbf{k}}{}^{\mathrm{f}_{k,1}^{-1}(l^{-1}(\vec{N}))}$ ssi $L \in \mathsf{REC}_k^{\vec{N}}$.*

Preuve : Prouvons ce résultat par induction sur $k \geq 1$.
Base : le cas $k = 1$ est obtenue directement du corollaire 4.3.23(2).
Pas d'induction : supposons le théorème valide pour $k \geq 1$. Le corollaire 4.3.23(1) et le théorème 5.2.4 assurent que la structure $\boldsymbol{\mathcal{P}}_{\mathbf{k}}{}^{\mathrm{f}_{k,1}^{-1}(l^{-1}(\vec{N}))}$ vérifie la propriété MD.

Étant donné $D \subseteq \mathcal{P}_{(k+1)}$, le théorème 4.3.20 s'applique, puisque f_k est SOM-inversible, et donc tout ensemble D est SOM-définissable dans $\boldsymbol{\mathcal{P}}_{\mathbf{k}}{}^{\mathrm{f}_{k+1,1}^{-1}(l^{-1}(\vec{N}))}$ ssi il existe un vecteur $\vec{L}$ d'ensembles SOM-définissables dans $\boldsymbol{\mathcal{P}}_{\mathbf{k}}{}^{\mathrm{f}_{k,1}^{-1}(l^{-1}(\vec{N}))}$ tel que $D \in \mathsf{REC}^{\vec{L}}$. En appliquant l'hypothèse d'induction à chaque L_i, nous obtenons que D est SOM-définissable dans $\boldsymbol{\mathcal{P}_{k+1}}$ ssi $D \in \mathsf{REC}_{k+1}^{\vec{N}}$.
□

Rapellons que nous conjecturons fortement que toute structure $\langle \mathbb{N}, +1, \vec{N}\rangle$ vérifie la propriété MD.

Chapitre 6

Ensembles reconnaissables de piles itérées

Nous montrons que la structure des k-piles admet une théorie SOM décidable, vérifie la propriété du Modèle Définissable et que les définissables sont exactement les ensembles générés par les automates à k-piles, contrôlés par des propriétés définissables. Pour cela nous prouvons tout d'abord que la structure $\mathbf{Pile_k}$ est équivalente (pour la logique SOM) à la structure $\boldsymbol{\mathcal{P}_k}$ étudiée dans le chapitre précédent.

6.1 Groupe libre versus piles itérées

Le résultat principal de cette section chapitre est que la structure $\mathbf{Pile_k}$ est SOM-équivalente à la structure $\boldsymbol{\mathcal{P}_k} = \langle \mathcal{P}_k, (\bullet_a)_{a \in \widehat{A_k}} \rangle$ où $\bullet_a$ représente le produit à droite par la lettre a dans le groupe libre $(\mathrm{Irr}(\mathcal{A}_k), \bullet, \varepsilon)$.

6.1.1 Expression des instructions classiques

Nous étudions ici les liens entre l'application d'une instruction à une k-pile et le produit à droite dans $\mathcal{P}_k$ (via la bijection $\varphi_k : k\text{-}\mathsf{Pile} \to \mathcal{P}_k$ introduite 1.4.5). Ces résultats seront utilisés pour comparer les structures $\boldsymbol{\mathcal{P}_k}$ et $\mathbf{Pile_k}$ dans la section 6.1.2, ainsi que dans la partie III, pour comparer différentes classes d'automates à k-piles.

Nous commençons par vérifier que pour tout $\omega \in k\text{-}\mathsf{Pile}$, le mot $\varphi_k(\omega)$ décrit une suite d'instructions permettant de calculer ω à partir de $\perp_k$. Si nous considérons par exemple, la 3-pile $\omega_{ex} = a_3[a_2[a_1a_1\ \perp]b_2[a_1\ \perp]\ \perp_2]\ \perp\ [a_2[a_1\ \perp]a_2[\perp]\ \perp_2]$, alors de mot $\varphi_3(\underline{\omega_{ex}}) = a_2a_2a_1a_3\bar{a_1}\bar{a_2}a_1a_2a_1$ code la suite d'instructions push_{2,a_2}, push_{2,a_2}, push_{1,a_1}, push_{3,a_3}, $\overline{\mathrm{push}}_{1,a_1}$, $\overline{\mathrm{push}}_{2,a_2}$, push_{1,a_1}, push_{2,a_2}.

Nous déterminons ensuite, pour tout instr dans $\mathfrak{I}nstr_k$, quel produit appliquer à $\varphi_k(\omega)$ pour obtenir $\varphi_k(\mathrm{instr}(\omega))$.

Le problème inverse est plus simple et découle immédiatement de la définition de φ_k :

Lemme 6.1.1. *Pour tous $u, v \in \mathcal{P}_k$, et $a \in A_i$, $\leq i \leq k$,*
- *$v = u \bullet a$ ssi ${\varphi_k}^{-1}(v) = \mathrm{push}_{i,a}({\varphi_k}^{-1}(u))$,*
- *$v = u \bullet \bar{a}$ ssi ${\varphi_k}^{-1}(u) = \mathrm{push}_{i,a}({\varphi_k}^{-1}(v))$ ssi ${\varphi_k}^{-1}(v) = \overline{\mathrm{push}}_{i,a}({\varphi_k}^{-1}(u))$.*

La figure 17 illustre ce résultat.

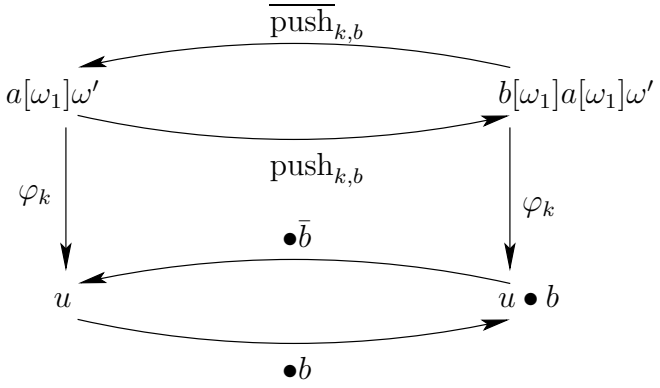


FIG. 17 – Correspondance instruction/produit à droite

Preuve : Ce résultat se vérifie par induction sur $k \geq 1$. Si $k = 1$, c'est évident. Supposons donc le lemme vrai pour $k \geq 1$. Soient $u, v \in \mathcal{P}_{k+1}$, et $\omega = a_{k+1}[\omega_1]\omega'$ tel que $\varphi_{k+1}(\omega) = u$,

- si $a \in A_{k+1}$, alors $u \bullet a = ua$ et par définition ${\varphi_{k+1}}^{-1}(u \bullet a) = a[\omega_1]\omega = \text{push}_{k+1,a}(\omega)$,
- si $a \in A_{k+1}$ alors $u \bullet \bar{a} = u'$ où $u = u' \cdot a$. Par définition,

$$\omega = \text{push}_{k+1,a}({\varphi_{k+1}}^{-1}(u')) = \text{push}_{k+1,a}({\varphi_{k+1}}^{-1}(u \bullet \bar{a}))$$

  En d'autres mots, ${\varphi_{k+1}}^{-1}(u \bullet \bar{a}) = \overline{\text{push}}_{k+1,a}(\omega)$.
- soit $a \in \widehat{A_{1,k}}$, alors $\varphi_{k+1}(u \bullet a) = a_{k+1}[\varphi_k(\text{f}_k(u \bullet a))]\omega'$ donc par hypothèse d'induction :
  - si $a \in A_i$, alors $\varphi_{k+1}(u \bullet a) = a_{k+1}[\text{push}_{i,a}(\omega_1)]\omega' = \text{push}_{i,a}(\omega)$,
  - si $a \in \overline{A_i}$, alors $\varphi_{k+1}(u \bullet a) = a_{k+1}[\overline{\text{push}}_{i,a}(\omega_1)]\omega' = \overline{\text{push}}_{i,a}(\omega)$.

□

Remarque 6.1.2. *Par définition de* $\overline{\text{push}}_{i,a}$ *le second point du lemme précédent et la remarque 1.4.11 impliquent que pour tout* $k \geq 1$, $u \in \mathcal{P}_k$, $a \in A_i$, $1 \leq i \leq k$,

$$u \bullet \bar{a} \in \mathcal{P}_k \text{ ssi } \text{top}_i(\varphi_k^{-1}(u)) = a \text{ et } \varphi_k^{-1}(u \bullet \bar{a}) = \text{pop}_i(\varphi_k^{-1}(u)).$$

Par contre, si $\text{top}_i(\omega) = a$, alors $\varphi_k(\text{pop}_i(\omega))$ n'est pas égal à $\varphi_k(\omega) \bullet \bar{a}$, mais au produit de $\varphi_k(\omega)$ avec un mot dans $\text{Irr}(A_{1,i-1}) \cdot \bar{a}$.

Proposition 6.1.3. *Pour tout* $\omega \in k\text{-Pile}(A_1, \ldots, A_k)$,

$$\exists \omega', \text{pop}_i(\omega) = \omega' \text{ ssi } \exists w \in \text{Irr}(A_{1,i-1}) \cdot \overline{A_i},\ \varphi_k(\omega) \bullet w \in \mathcal{P}_k.$$

Dans ce cas

$$\varphi_k(\omega) \bullet w = \varphi_k(\text{pop}_i(\omega))$$

et $w = \overline{u} \cdot \bar{a}$ *où* $a = \text{top}_i(\omega)$ *et* $\text{f}_{k,i}(u) \in \mathcal{P}_i \cdot a \cdot u$.

Preuve : Si $i = k \geq 1$, alors

$$\begin{array}{lll} \exists \omega',\ \text{pop}_k(\omega) = \omega' & \text{ssi} & \exists \omega' \in (k-1)\text{-Pile}, \exists a \in A_k, \omega = a[\omega_1]\omega' \\ & \text{ssi} & \exists \omega', a \in A_k, \varphi_k(\omega) = \varphi_k(\omega') \cdot a \cdot \text{f}_{k-1}(\varphi_{k+1}(\omega')) \bullet \varphi_{k-1}(\omega_1) \\ & \text{ssi} & \exists w = ua,\ a = \text{top}_k(\omega), \varphi_k(\omega) \in \mathcal{P}_k \cdot a \cdot u,\ \varphi_k(\omega') = \varphi_k(\omega) \cdot \overline{w}. \end{array}$$

Sinon, la proposition se vérifie par induction sur $k \geq 1$. Supposons la propriété vérifiée pour tout k, et considérons $\omega = a[\omega_1]\omega'' \in (k+1)$-Pile, et $0 \leq i \leq k$.

$$\begin{aligned}\exists \omega',\ \mathrm{pop}_i(\omega) = \omega' \quad &\text{ssi} \quad \exists \omega_1',\ \omega' = a[\omega_1']\omega'' \text{ et } \mathrm{pop}_i(\omega_1) = \omega_1' \\ &\text{ssi} \quad \exists \omega_1',\ \varphi_{k+1}(\omega') = \varphi_{k+1}(\omega) \bullet \overline{\mathrm{f}_k(\varphi_k(\omega_1))} \bullet \mathrm{f}_k(\varphi_k(\omega_1')).\end{aligned}$$

En appliquant l'hypothèse d'induction, nous obtenons

$$\begin{aligned}\exists \omega',\ \mathrm{pop}_i(\omega) = \omega' \quad &\text{ssi} \quad \exists w \in \mathrm{Irr}(A_{1,i})\overline{A_i},\ \varphi_{k+1}(\omega') = \varphi_{k+1}(\omega) \bullet \overline{\mathrm{f}_k(\varphi_k(\omega_1))} \bullet \mathrm{f}_k(\varphi_k(\omega_1)) \bullet w \\ &\text{ssi} \quad \exists w \in \mathrm{Irr}(A_{1,i})\overline{A_i},\ \varphi_{k+1}(\omega') = \varphi_{k+1}(\omega) \bullet w.\end{aligned}$$

De la même façon, en appliquant l'hypothèse d'induction, nous obtenons la seconde équivalence :

$$\exists \omega',\ \mathrm{pop}_i(\omega) = \omega' \quad \text{ssi} \quad \exists w = ua,\ a = \mathrm{top}_k(\omega), \mathrm{f}_{k,i}(\varphi_k(\omega)) \in \mathcal{P}_i \cdot a \cdot u,\ \varphi_k(\omega') = \varphi_k(\omega) \cdot \overline{w}.$$

□

De façon similaire, l'application d'une instruction change se traduit dans $\mathcal{P}_k$ par un produit à droite.

Proposition 6.1.4. *Pour tout $\omega \in k$-Pile$(A_1, \ldots, A_k)$, $i \in [1,k]$, $a \in A_i$,*

$$\exists \omega', \mathrm{change}_a(\omega) = \omega' \textit{ ssi } \exists w \in \mathrm{Irr}(A_{1,i-1}) \textit{ et } b \in A_i,\ \varphi_k(\omega) \bullet w\bar{b} \in \mathcal{P}_k.$$

Dans ce cas,

$$\varphi_k(\mathrm{change}_a(\omega)) = \varphi_k(\omega) \bullet w\bar{b} \bullet a\overline{w}.$$

Preuve : La preuve de cette proposition suit les mêmes lignes que celle de la proposition 6.1.3.

□

6.1.2 Équivalence avec la logique sur les k-piles

Nous utilisons les résultats obtenus dans la section 6.1.1 afin de comparer (pour la logique SOM) les structures

$$\boldsymbol{\mathcal{P}_k} = \langle \mathcal{P}_k,\ \varepsilon,\ (\bullet_a)_{a \in \widehat{A_{1,k}}} \rangle \text{ et } \mathbf{Pile_k} = \langle k\text{-Pile},\ \perp_k,\ (\mathrm{PUSH}_a, \mathrm{POP}_i, \mathrm{CHANGE})_{i \in [1,k], a \in A_i} \rangle.$$

Le lemme 6.1.1 permet d'établir immédiatement que la structure $\boldsymbol{\mathcal{P}_k}$ est SOM-interprétable dans $\mathbf{Pile_k}$.

Proposition 6.1.5. *L'application $\varphi_k{}^{-1} : \boldsymbol{\mathcal{P}_k} \to \mathbf{Pile_k}$ est une* SOM*-interprétation.*

Prouvons maintenant que φ_k est une SOM-interprétation de $\mathbf{Pile_k}$ dans $\boldsymbol{\mathcal{P}_k}$. Pour cela, nous devons être capable, pour toute instruction instr $\in \mathfrak{I}nstr_k$, de trouver une SOM-formule $\Phi_{\mathrm{instr}}(x,y)$ satisfaite dans $\boldsymbol{\mathcal{P}_k}$ par exactement tous les couples $(\varphi_k(\omega), \varphi_k(\mathrm{instr}(\omega)))$.

Les instructions composant $\mathfrak{I}nstr_k$ sont particulières puisque se sont toutes des instructions de niveau i (ou i-instructions), pour $i \in [1,k]$ (voir §1.3). Une i-instruction a la particularité d'être complètement définie par sa valeur sur i-Pile. En utilisant cette propriété, il possible de prouver simplement l'existence d'une formule $\Phi_{\mathrm{instr}}(x,y)$ comme ci-dessus.

Soit instr une instruction de niveau k quelconque, le lemme suivant établi que pour tout $i \geq 0$, pour prouver que instr est SOM-définissable dans $\boldsymbol{\mathcal{P}_{k+i}}$, il suffit de démontrer qu'elle est SOM-définissable dans $\boldsymbol{\mathcal{P}_k}$.

Lemme 6.1.6. *Soit* instr *une instruction de niveau* $k \geq 1$ *sur les alphabets* $A_1, \ldots, A_k$, *et* $\Phi(x, y)$ *une formule dans* $\mathrm{SOM}(Sig_k)$, *vérifiant pour tous* $u, v \in \mathcal{P}_k$,

$$\boldsymbol{\mathcal{P}_k} \models \Phi(u, v) \text{ ssi } {\varphi_k}^{-1}(v) = \mathrm{instr}({\varphi_k}^{-1}(u)),$$

alors, pour tout $i \geq 0$, *il existe une formule* $\Phi_{+i}(x, y) \in \mathrm{SOM}(Sig_{k+i})$ *telle que pour tous* $u, v \in \mathcal{P}_{k+i}$,

$$\boldsymbol{\mathcal{P}_{k+i}} \models \Phi_{+i}(u, v) \text{ ssi } {\varphi_{k+i}}^{-1}(v) = \mathrm{instr}({\varphi_{k+i}}^{-1}(u)).$$

Preuve : La proposition 1.4.7 et la définition de ${\varphi_k}^{-1}$, impliquent que pour tous $v, v' \in (k+1)$-Pile et $\omega_1, \omega_1' \in k$-Pile, les propriétés suivantes sont équivalentes :

1. il existe $\omega \in (k+1)\text{-Pile} \cup \{\varepsilon\}$ et $a \in A_{k+1} \cup \{\bot\}$ tel que
   $${\varphi_{k+1}}^{-1}(v) = a[\omega_1]\omega \text{ et } {\varphi_{k+1}}^{-1}(v') = a[\omega_1']\omega,$$
2. il existe $u \in \mathrm{Irr}(A_k)$ tel que $v' = v \bullet u$ et
   $$\mathrm{f}_k(v) = \varphi_k(\omega_1), \quad \mathrm{f}_k(v') = \varphi_k(\omega_1').$$

Alors, si Φ vérifie $\boldsymbol{\mathcal{P}_k} \models \Phi(u, v)$ ssi ${\varphi_k}^{-1}(v) = \mathrm{instr}({\varphi_k}^{-1}(u))$, en utilisant la proposition 5.1.3, nous obtenons une formule Φ_{+i} satisfaisant le lemme par la construction inductive suivante :

- $\Phi_{+0}(x, y) := \Phi(x, y)$
- $\forall i \geq 0$, $\Phi_{+(i+1)}(x, y) := \exists u \in \widehat{A_{1,k+i}}^*$, $y = x \bullet u \wedge (\Phi_{+i})^{+1}(x, y)$.

□

Théorème 6.1.7. *Pour tout* $k \geq 1$, *pour tous alphabets de piles* $A_1, \ldots, A_k$, *l'application* $\varphi_k : \mathbf{Pile_k}(A_1, \ldots, A_k) \to \boldsymbol{\mathcal{P}_k}(A_1, \ldots, A_k)$ *est une SOM-interprétation.*

Preuve : En utilisant le lemme précédent, il suffit de montrer que $\mathrm{push}_{k,a}$, pop_k et $\mathrm{change}_{k,a}$ sont SOM-définissable dans $\boldsymbol{\mathcal{P}_k}$. En utilisant les propositions 6.1.1 pour l'instruction push, 6.1.3 pour l'instruction pop et 6.1.4 pour l'instruction change, il est clair que pour tout $a \in A_k$, les formules suivantes satisfont le théorème.

- $\mathrm{PUSH}_{k,a}(x, y) := \ y = x \bullet a$,
- $\mathrm{POP}_k(x, y) := \exists a \in A_k,\ w \in \mathrm{Irr}(A_{k-1}),\ x = y \bullet a \bullet w$,
- $\mathrm{CHANGE}_{k,a}(x, y) := \exists u, u' \in \mathrm{Irr}(A_{k-1}),\ \exists b \in A_k,\ y = x \bullet u' \bullet \bar{b} \bullet a \bullet u \wedge x\,(=)^{+1}\,y$.

□

Les structures $\boldsymbol{\mathcal{P}_k}$ et $\mathbf{Pile_k}$ ont donc des logiques SOM "isomorphes". Rappelons que la classe des ensembles définissables dans $\mathbf{Pile_k}(A_1, \ldots, A_k)^{\vec{C}}$ est notée $\mathsf{DEF}_k^{\vec{C}}(A_1, \ldots, A_k)$.

Proposition 6.1.8. *Pour tout* $D \in k\text{-Pile}(A_1, \ldots, A_k)$,

$$D \in \mathsf{DEF}_k^{\vec{C}}(A_1, \ldots, A_k) \text{ ssi } \varphi_k(D) \text{ est définissable dans} \boldsymbol{\mathcal{P}_k}^{\varphi_k(\vec{C})}$$

En particulier,

$$D \in \mathsf{DEF}_k^{\vec{C}}(A_1, \ldots, A_k) \text{ ssi } \varphi_k(D) \in \mathsf{REC}_k(A_1, \ldots, A_k).$$

6.2 Ensembles définissables de piles itérées

L'équivalence entre les structures $\boldsymbol{\mathcal{P}_k}$ et $\mathbf{Pile_k}$ que nous venons de prouver nous permettent de translater vers $\mathbf{Pile_k}$ les résultats obtenus dans la section 5 sur les propriétés SOM de la structure.

Théorème 6.2.1. *Pour tout $k \geq 1$, la structure* $\mathbf{Pile_k}$ *admet une* SOM*-théorie et vérifie la propriété* MD*.*

Preuve : Puisque d'après le théorème 6.1.7, la structure $\mathbf{Pile_k}$ est interprétable dans $\boldsymbol{\mathcal{P}}_\mathbf{k}$ et d'après le théorème 5.2.3 la structure $\boldsymbol{\mathcal{P}}_\mathbf{k}$ admet une SOM-théorie décidable et vérifie MD, nous obtenons :

1. $\mathbf{Pile_k}$ admet une SOM-théorie décidable (d'après le théorème 3.1.4)
2. Pour toute formule $\phi(X) \in \mathrm{SOM}(\mathbf{Pile_k})$, il existe une $\psi(X) \in \mathrm{SOM}(\boldsymbol{\mathcal{P}}_\mathbf{k})$, dont les modèles sont exactement les images par φ_k des modèles de ϕ. Par MD, il existe un modèle ${\varphi_k}^{-1}(D)$ de ψ qui est SOM-définissable dans $\boldsymbol{\mathcal{P}}_\mathbf{k}$. Donc D est un modèle de ϕ et puisque d'après la proposition 6.1.5, φ_k^{-1} est une interprétation de $\boldsymbol{\mathcal{P}}_\mathbf{k}$ dans $\mathbf{Pile_k}$, le modèle D est définissable dans $\mathbf{Pile_k}$. La structure $\mathbf{Pile_k}$ vérifie donc la propriété MD.

□

Théorème 6.2.2. *Soit $\vec{R}$ un vecteur de sous-ensembles de A_1^*,*

1. *si la* SOM*-théorie de $\langle {A_1}^*, (\mathrm{SUCC}_a)_{a\in A_1}, \vec{R}\rangle$ est décidable, alors pour tout $k \geq 1$, la structure* ${\mathbf{Pile_k}}^{{\mathrm{p}_{k,1}}^{-1}(\vec{R})}$ *admet une* SOM*-théorie décidable,*
2. *si $\langle A_1^*, (\mathrm{SUCC}_a)_{a\in A_1}, \vec{R}\rangle$ vérifie la propriété* MD*, alors* ${\mathbf{Pile_k}}^{{\mathrm{p}_{k,1}}^{-1}(\vec{R})}$ *vérifie également la propriété* MD*.*

Preuve : Ce théorème découle directement du théorème 5.2.4, de la SOM-équivalence des structures $\boldsymbol{\mathcal{P}}_\mathbf{k}$ et $\mathbf{Pile_k}$, et par la correspondance existant entre les projection p_k et f_k via les bijections φ_{k+1} et φ_k (voir proposition 1.4.9).

□

Définition 6.2.3. *Pour tout $k \geq 0$, nous notons* ${\mathsf{PREC}_k}^{\vec{N}}$ *la classe des ensembles $D \in k$-*Pile *tels que $\varphi_k(D) \in \mathsf{REC}_k^{\vec{N}}$.*

Proposition 6.2.4. *Pour tout $k \geq 1$, et tout vecteur d'ensembles d'entiers $\vec{N}$ tel que la structure $\langle \mathbb{N}, +1, \vec{N}\rangle$ vérifie la propriété* MD*,*

${\mathsf{PREC}_k}^{\vec{N}}$ *est la classe des ensembles définissables en* SOM *dans* ${\mathbf{Pile_k}}^{{\mathrm{f}_k}^{-1}(l^{-1}(\vec{N}))}$.

Preuve : Ce résultat est évident en utilisant la proposition 6.1.8.

□

Proposition 6.2.5. *Pour tout $k \geq 1$, et tout vecteur d'ensembles d'entiers $\vec{N}$ tel que la structure $\langle \mathbb{N}, +1, \vec{N}\rangle$ vérifie la propriété* MD*,*

1. $\forall D \subseteq k$-$\mathsf{Pile}^{\vec{N}}$*, si* $D \in {\mathsf{PREC}_k}^{\vec{N}}$ *alors* ${\mathrm{p}_{k+1,k}}^{-1}(D) \in {\mathsf{PREC}_{k+1}}^{\vec{N}}$*,*
2. $\forall D \subseteq (k+1)$-$\mathsf{Pile}^{\vec{N}}(A_1, \ldots, A_k)$*, si* $D \in \mathsf{PREC}_{k+1}$ *alors* $\mathrm{p}_k(D) \in {\mathsf{PREC}_k}^{\vec{N}}$*.*

Preuve :

1. Si $D \in {\mathsf{PREC}_k}^{\vec{N}}$, alors par définition, $\varphi_k(D) \in \mathsf{REC}_k^{\vec{N}}$, et en utilisant la proposition 5.1.4 ${f_k}^{-1}(\varphi_k(D)) \in \mathsf{REC}_{k+1}^{\vec{N}}$, c'est-à-dire, d'après la proposition 1.4.7, ${\mathrm{p}_{k+1,k}}^{-1}(D) \in {\mathsf{PREC}_{k+1}}^{\vec{N}}$.

2. Si $D \in \mathsf{PREC}_{k+1}{}^{\vec{N}}$ alors en utilisant encore la proposition 1.4.7, pour prouver que $\mathrm{p}_k(D) \in \mathsf{PREC}_k{}^{\vec{N}}$, il suffit de vérifier que $\mathrm{f}_k(\varphi_{k+1}(D)) \in \mathsf{REC}_k^{\vec{N}}$.

   Posons $R = \varphi_{k+1}(D)$. Cet ensemble appartient à $\mathsf{REC}_{k+1}^{\vec{N}}$ et par définition, R est reconnu par un automate fini $\mathcal{A}$ avec oracles $\mathrm{f}_k{}^{-1}(\vec{O})$, où $\vec{O}$ est un vecteur le langages dans $\mathsf{REC}_k^{\vec{N}}$. Notons Q l'ensemble des états de $\mathcal{A}$, et pour tout $q \in Q$, désignons par C_q l'ensemble des mots ayant un calcul dans $\mathcal{A}$ aboutissant en q.
   Par définition de $\mathcal{P}_{k+1}$, pour tout $u \in \mathcal{P}_{k+1}$, pour tout $a \in \widehat{A_{1,k}}$,

$$ua \in \mathcal{P}_{k+1} \text{ ssi } \boldsymbol{\mathcal{P}_k} \models \bullet_a(\mathrm{f}_k(u), \mathrm{f}_k(ua))$$

   Alors, $(\mathrm{f}_k(C_q))_{q\in Q}$ est la famille formée des plus petits $S_q \subseteq \mathcal{P}_k$, $q \in Q$ vérifiant :
   $\forall q \in Q$, $u \in \mathcal{P}_k$,
   $u \in S_q$ ssi
   – soit $q = q_0$ et $u = \varepsilon$,
   – soit il existe $(p, a, \vec{o}, q) \in \Delta$ et $v \in S_p$ tels que $\chi_{\vec{O}}(v) = \vec{o}$ et
     si $a \in \widehat{A_{1,k}}$ alors $\bullet_a(v, u)$ sinon (cas où $a \in A_{k+1}$) $u = v$.

   Puisque d'après le théorème 5.2.9, $\vec{O}$ est définissable dans $\boldsymbol{\mathcal{P}_k}^{\mathrm{f}_{k,1}{}^{-1}(l^{-1}(\vec{N}))}$, chacun des $\mathrm{f}_k(C_q)$ est donc également définissable dans $\boldsymbol{\mathcal{P}_k}^{\mathrm{f}_{k,1}{}^{-1}(l^{-1}(\vec{N}))}$, et il est alors simple de construire une formule définissant $\mathrm{f}_k(R)$ dans $\boldsymbol{\mathcal{P}_k}^{\mathrm{f}_{k,1}{}^{-1}(l^{-1}(\vec{N}))}$.
   Donc $\mathrm{f}_k(R) \in \mathsf{REC}_k^{\vec{N}}$, c'est à dire $\mathrm{p}_k(D) \in \mathsf{PREC}_k{}^{\vec{N}}$.

□

Proposition 6.2.6. *Pour tout $k \geq 1$, et tout vecteur d'ensembles d'entiers $\vec{N}$ tel que la structure $\langle \mathbb{N}, +1, \vec{N}\rangle$ vérifie la propriété* MD,

$\mathsf{PREC}_{k+1}{}^{\vec{N}}$ *est l'union des classes* $\mathsf{ACC}_{k+1}{}^{\mathrm{p}_k{}^{-1}(\vec{R})}$, *pour $\vec{R}$ vecteur d'éléments de* $\mathsf{PREC}_k{}^{\vec{N}}$.

Preuve : Soit $P = \varphi_{k+1}^{-1}(L) \in \mathsf{PREC}_{k+1}{}^{\vec{N}}$, pour $k \geq 0$. Par définition, il existe un automate $\mathcal{A} = (Q, \mathrm{f}_k{}^{-1}(\vec{O}), \Delta, q_0, F) \in \mathsf{AF}_{k+1}{}^{\vec{N}}(A_1, \ldots, A_k)$ reconnaissant L et $\vec{O}$ est un vecteur de langages dans $\mathsf{REC}_k\vec{N}$.

Nous pouvons sans perte de généralité supposer que $\mathcal{A}$ est déterministe (théorème 4.1.8), et complet dans $\mathcal{P}_{k+1}$, (i.e., pour tout $u \in \widehat{A_{1,k+1}}^*$, u admet un calcul dans $\mathcal{A}$ ssi $u \in \mathcal{P}_{k+1}$).

Considérons le $(k+1)$-AP suivant : $\mathcal{A} = (Q, \widehat{A_{1,k+1}}, (A_1, \ldots, A_{k+1}), \mathrm{p}_k{}^{-1}(\vec{R}), \Delta', q_0, \perp)$, avec $\vec{R} = \varphi_k{}^{-1}(\vec{O}) \in \mathsf{PREC}_k{}^{\vec{N}}$, et Δ' construit de la façon suivante :

– $\forall (p, a, \vec{o}, q) \in \Delta$, $a \in A_i$, $1 \leq i \leq k+1$, posons $\forall w \in \mathrm{top}((k+1)\text{-}\mathsf{Pile}(A_1, \ldots, A_{k+1}))$

$$(p, a, w, \vec{o}, \mathrm{push}_{i,a}, q) \in \Delta',$$

– $\forall (p, \bar{a}, \vec{o}, q) \in \Delta$, $a \in A_i$, $1 \leq i \leq k$, posons $\forall w \in \mathrm{top}((k+1)\text{-}\mathsf{Pile}(A_1, \ldots, A_{k+1}))$

$$(p, \bar{a}, w, \vec{o}, \mathrm{pop}_i, q) \in \Delta'.$$

En utilisant le lemme 6.1.1, il apparaît clairement que tout calcul dans $\mathcal{A}$ est de la forme : $(q_0, \varphi_{k+1}{}^{-1}(\omega), \perp_{k+1}) \vdash_{\mathcal{A}}^* (p, \varepsilon, \omega)$ pour p et ω tels que

$$(q_0, \uparrow \varphi_{k+1}(\omega)) \rightarrow_{\mathcal{A}}^* (p, \varphi_{k+1}(\omega)\uparrow)$$

Donc $\varphi_{k+1}(\mathrm{ACC}(\mathcal{A})) = \mathrm{L}(\mathcal{A})$, ce qui démontre que $P \in \mathsf{ACC}_{k+1}{}^{\mathrm{p}_k{}^{-1}(\vec{R})}$, pour $\vec{R}$ vecteur d'éléments de $\mathsf{PREC}_k{}^{\vec{N}}$.

Inversement, considérons $P \in \mathsf{ACC}_{k+1}{}^{\mathrm{p}_k{}^{-1}(\vec{R}}$, pour $\vec{R}$ composé d'éléments de $\mathsf{PREC}_k{}^{\vec{N}}$. D'après la proposition 6.2.5, $\mathrm{p}_k{}^{-1}(\vec{R})$ est composé d'éléments de $\mathsf{PREC}_{k+1}{}^{\vec{N}}$, et donc définissables dans $\mathbf{Pile_{k+1}}{}^{\mathrm{f}_{k+1,1}{}^{-1}(l^{-1}(\vec{N}))}$. D'après la proposition 3.3.6, l'ensemble d'accessibles P est donc également définissable dans cette structure, c'est à dire, $P \in \mathsf{PREC}_{k+1}{}^{\vec{N}}$.

□

Puisque la structure $\langle , \mathbb{N}, +1\rangle$ satisfait la propriété MD, en nous restreignant au cas où $\vec{N}$ est le vecteur nul, nous obtenons le théorème général suivant.

Théorème 6.2.7. *Pour tout $S \subseteq k$-$\mathsf{Pile}(A_1, \ldots, A_k)$, $k \geq 1$, les propriétés suivantes sont équivalentes :*

1. *S est définissable dans $\mathbf{Pile_k}(A_1, \ldots, A_k)$, (i.e. $S \in \mathsf{DEF}_k(A_1, \ldots, A_k)$*
2. *il existe un vecteur $\vec{C}$ de sous ensembles de PREC_{k-1} tel que S est engendré par un automate dans k-$\mathsf{AP}(A_1, \ldots, A_k)^{\mathrm{p}_{k-1}{}^{-1}(\vec{C})}$,*
3. *$\varphi_k(S)$ est définissable dans $\boldsymbol{\mathcal{P}}_{\mathbf{k}}(A_1, \ldots, A_k)$*
4. *$\varphi_k(S)$ est reconnu par un automate dans $\mathsf{AF}_k(A_1, \ldots, A_k)$, (i.e., $\varphi_k(S) \in \mathsf{REC}_k$),*
5. *$S \in \mathsf{PREC}_k$.*

Preuve : L'équivalence entre les points (1),(4) et (5) est prouvé par la proposition 6.1.8, l'équivalence entre (3) et (4) est énoncée dans le théorème 5.2.7, l'équivalence entre (2) et (5) est démontré dans la proposition 6.2.6

□

Un théorème général similaire peut-être énoncé pour la classe $\mathsf{PREC}_k{}^{\vec{N}}$, sous la condition : $\langle \mathbb{N}, +1, \vec{N}\rangle$ vérifie MD.

Liens avec d'autres travaux Dans [Car05], l'auteur introduit une notion d'ensembles de piles k-réguliers. Il étudie une classe Reg_k correspondant à l'ensemble des piles accessibles par un automate à k-piles utilisant exclusivement des instructions du type push et $\overline{\text{push}}$. Il donne une représentation normalisée des éléments de cette classe sous forme d'expression régulière dans un monoïde sur $(\widehat{A_{1,k}} \cup T_k)^*$ où, T_k est un alphabet infini composé de tous les T_R, pour $R \in Reg_{k-1}$. Cette normalisation généralise les résultats obtenus dans [Cau96] pour le niveau 1, et permet de prouver que Reg_k forme une algèbre booléenne effective. Il prouve également que la classe Reg_k correspond (à codage près) aux ensembles définissables dans la structure $\mathbf{Pile_k}$, ce qui montre que la classe Reg_k coïncide avec la classe PREC_k définie dans ce chapitre.

L'union de ces deux travaux disjoints montre que les classe PREC_k bénéficient de nombreuses propriétés généralisant celles de la classe PREC_1 (qui correspond à isomorphisme près à la classe REC des langages réguliers). De plus, la représentation des piles par des mots dans le groupe libre semble être complètement adaptée à la généralisation de la notion de langages réguliers.

Troisième partie

Étude des automates à piles itérées

Introduction

Nous explorons dans cette partie les différentes classes d'automates et de machines permettant de reconnaître les langages de niveaux supérieurs. Nous définissons pour cela la notion de *simulation déterministe* d'une machine par une autre. L'extension de cette relation aux classes de machines permet de comparer, non seulement les classes de langages reconnues, mais également les classes de langages **déterministes** et les ensembles de piles générés. Tous les résultats de comparaison de cette partie sont établis pour cette relation.

Nous établissons dans un premier temps, l'équivalence entre la classe des k-AP et chacune des différentes classes d'automates normalisés introduites §2.3.1. Nous montrons que les restrictions de ces classes à des contrôleurs nuls ou à des contrôleurs réguliers (i.e., définissables en LSOM) sont également équivalentes. Ces résultats sont résumés dans le théorème 8.2.4.

Dans chapitre suivant, nous étudions les équivalences entre k-AP contrôlés et it-AP non contrôlés. Nous nous intéressons à deux types particuliers de contrôleurs :

Contrôleur d'ordre i : Soit C un contrôleur de niveau i, (i.e., inclus dans i-Pile), nous notons $[C]_{i,k}$ la classe des $\omega \in k$-Pile tels que $\mathrm{p}_i(\omega) \in C$. Pour une machine contrôlée par une vecteur $[\vec{C}]_{i,k}$, les tests ne portent donc que sur l'appartenance à chaque C_j de la projection de la mémoire dans i-Pile.

Contrôleur du mot de niveau i : Soit L un langage sur un alphabet de niveau i, nous notons $(L)_{i,k}$ la classe des $\omega \in k$-Pile tels que $\mathrm{mot}_i(\omega) \in L$. Il s'agit donc d'un contrôleur de niveau i particulier. Pour une machine contrôlée par un vecteur $[\vec{C}]_{i,k}$, les tests ne portent donc que sur l'appartenance à chaque C_j du mot de niveau i de la mémoire.

Nous prouvons l'équivalence de la classe des k-AP non-contrôlés avec les classes suivantes :
- k-AP à instructions symétriques (voir définition 2.3.2) non contrôlés (théorème 9.3.8),
- k-AP à contrôleurs $[C]_{i,k}$, pour $i \in [1,k]$ et C définissable en LSOM (théorème 9.3.7),
- $(k-\ell)$-AP (pour $1 \leq \ell < k$) contrôlés par des vecteurs du type $(\vec{L})_{1,k-\ell}$, où les composantes de $\vec{L}$ sont toutes reconnues par un même ℓ-AP non-contrôlé déterministe (pour des ensembles d'états finaux différents) (théorème 9.3.11).

Nous terminons cette partie en développant dans le dernier chapitre différentes applications de ces résultats. Nous établissons en particulier deux nouvelles caractérisations des langages de niveau k : comme projection des langages $(k+1)$-réguliers, comme cône rationnel engendré par le langage $\mathcal{M}_k$ codant les mouvements valides de piles (voir définition (9.1.10).

Chapitre 7

Comparaisons de machines

Afin de comparer les machines entres elles, nous désirons prendre en compte, les langages acceptés, mais également les autres paramètres plus internes des machines. Nous introduisons pour cela la notion de *simulation déterministe* d'une machine par une autre. Les principales caractéristiques de la simulation déterministes sont la préservation du langage reconnu, du déterminisme, de la "structure" du graphe de calcul et de l'ensemble de piles généré. Cette notion permettra donc d'étudier finement les similitudes entre les machines.

7.1 Simulation déterministe

Nous considérons dans toute cette sous-section les deux machines quelconques suivantes : $\mathcal{M}_1 = (\vec{C}, Q_1, \Sigma, (A_1, \ldots, A_k), \Delta_1, q_0, \bot, F_1)$ et $\mathcal{M}_2 = (\vec{D}, Q_1 \cup Q_2, \Sigma, (B_1, \ldots, B_\ell), \Delta_2, q_0, \bot', F_2)$, avec $Q_1 \cap Q_2 = \emptyset$.

Considérons une fonction injective $\mathcal{S} : k\text{-}\mathsf{Pile}(A_1, \ldots, A_k) \to \ell\text{-}\mathsf{Pile}(B_1, \ldots, B_\ell)$. De façon intuitive, $\mathcal{S}$ est une *simulation déterministe* de $\mathcal{M}_1$ par $\mathcal{M}_2$, si le graphe de $\mathcal{M}_2$ est obtenu en substituant chaque arc $E_\alpha((p, \omega), (p', \omega'))$ du graphe de $\mathcal{M}_1$, par un chemin déterministe $(p, \mathcal{S}(\omega)), \ldots, (p', \mathcal{S}(\omega'))$ étiqueté par α, et dont tous les états des sommets internes appartiennent à Q_2.

Commençons par définir la notion de simulation déterministe d'un arc du graphe de calcul de $\mathcal{M}_1$. Chaque arc est simulé par les chemins obtenus par un système dit *standard*.

Définition 7.1.1 (Système de transitions standard). *Le système de transition Δ d'une machine quelconque est dit* (p, α, q)-standard *si il est* **déterministe** *et*

- *il n'y a aucune transition aboutissant en p,*
- *il n'y a aucune transition partant de q,*
- *toute transition issue de p lit la lettre α sur la bande d'entrée*

Définition 7.1.2 (Simulation déterministe d'un arc). *Supposons que $\Delta_1 = \{\delta_1, \ldots, \delta_n\}$, (et les δ_i sont distinctes). Soit $\mathcal{S} : k\text{-}\mathsf{Pile}(A_1, \ldots, A_k) \to \ell\text{-}\mathsf{Pile}(B_1, \ldots, B_\ell)$ une fonction injective, $\delta = (p, \alpha, w, \vec{\sigma}, q)$ une transition de $\mathcal{M}_1$, et $\Delta_{2,\delta}$ un ensemble de transitions inclus dans Δ_2 (la relation de transitions de $\mathcal{M}_2$).*

Soit $\omega \in k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$ appartenant au domaine de $\mathcal{S}$, nous notons ${\delta^{\vec{C}}} \leq_{\mathcal{S},\omega} {\Delta_{2,\delta}}^{\vec{D}}$ ssi

- *$\Delta_{2,\delta}$ est (p, α, q)-standard,*
- *$(p, \alpha, \omega) \vdash^1_{\delta^{\vec{C}}} (q, \varepsilon, \mathrm{instr}(\omega))$ ssi $(p, \alpha, \mathcal{S}(\omega)) \vdash^+_{{\Delta_{2,\delta}}^{\vec{D}}} (q, \varepsilon, \mathcal{S}(\mathrm{instr}(\omega)))$,*

– $\delta^{\vec{C}}$ *est applicable à l'état total* (p,ω) *ssi* ${\Delta_{2,\delta}}^{\vec{D}}$ *est applicable à* $(p,\mathcal{S}(\omega))$.

Remarque 7.1.3. *Dans la définition ci-dessus, "*${\Delta_{2,\delta}}^{\vec{D}}$ *est applicable à* $(p,\mathcal{S}(\omega))$*" signifie qu'il existe un état total quelconque* (q',ω') *tel que* $(p,\mathcal{S}(\omega)) \twoheadrightarrow^{+}_{{\Delta_{2,\delta}}^{\vec{D}}} (q',\omega')$.

Définition 7.1.4 (Simulation déterministe). *Une simulation déterministe de* $\mathcal{M}_1$ *par* $\mathcal{M}_2$ *est une application* $\mathcal{S} : \mathrm{ACC}_Q(\mathcal{M}_1) \to \mathrm{ACC}_Q(\mathcal{M}_2)$ **injective** *vérifiant* $\mathcal{S}(\bot_k) = \mathcal{S}(\bot'_\ell)$ *et* $\mathcal{S}(F_1) = F_2$ *et telle que :*

– *il existe une partition* $(Q_{2,i})_{i\in[1,n]}$ *de* Q_2*, et*

– *il existe une partition* $(\Delta_{2,i})_{i\in[1,n]}$ *de* Δ_2*,*

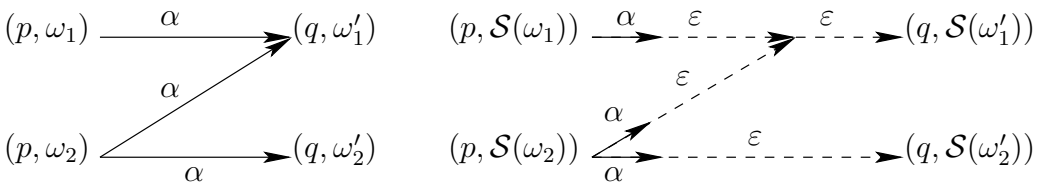
telles que pour toute transition $\delta_i = (p,\alpha,w,\vec{\sigma},q)$*,* $i \in [1,n]$*,*

– $\Delta_{2,i}$ *est défini sur* $Q_{2,i} \cup \{p,q\}$*,*

– *pour tout* $\omega \in \mathrm{ACC}_Q(\mathcal{M}_1)$*,* ${\delta_i}^{\vec{C}} \leq_{\mathcal{S},\omega} {\Delta_{2,i}}^{\vec{D}}$.

Remarque 7.1.5. *Le fait de choisir disjoints les systèmes de transitions et les ensembles d'états internes les composant permet d'assurer que tous les chemins du graphe de* $\mathcal{M}_2$ *simulant une transition de* $\mathcal{M}_1$ *sont déterministes.*

Le schéma 18 représente les graphes de calculs de $\mathcal{M}_1$ et $\mathcal{M}_2$ restreints respectivement aux arcs correspondant aux applications de $\delta^{\vec{C}}$ et $\Delta^{\vec{D}}_{2,\delta}$. Nous fixons les notations suivantes :



graphe de $\mathcal{M}_1$ graphe de $\mathcal{M}_2$

FIG. 18 – Simulation déterministe de $\mathcal{M}_1$ par $\mathcal{M}_2$

– nous notons $\mathcal{M}_1 \leq_{\mathcal{S}} \mathcal{M}_2$ ssi $\mathcal{S}$ est une simulation déterministe de $\mathcal{M}_1$ par $\mathcal{M}_2$,

– la notion de simulation peut être étendue aux familles de machines : si $\mathcal{F}_1 \subseteq k\text{-}\mathsf{MP}(A_1,\ldots,A_k)$ et $\mathcal{F}_2 \subseteq \ell\text{-}\mathsf{MP}(B_1,\ldots,B_\ell)$, nous notons $\mathcal{F}_1 \sqsubseteq_{\mathcal{S}} \mathcal{F}_2$ ssi pour tout $\mathcal{M}_1 \in \mathcal{F}_1$, il existe $\mathcal{M}_2 \in \mathcal{F}_2$ tel que $\mathcal{M}_1 \leq_{\mathcal{S}} \mathcal{M}_2$,

– nous notons $\mathcal{F}_1 \equiv \mathcal{F}_2$ ssi il existe $\mathcal{S}_1$ et $\mathcal{S}_2$ tels que $\mathcal{F}_1 \sqsubseteq_{\mathcal{S}_1} \mathcal{F}_2$ et $\mathcal{F}_2 \sqsubseteq_{\mathcal{S}_2} \mathcal{F}_1$.

Remarque 7.1.6.

1. *Si* $\mathcal{M}_1 \leq_{\mathcal{S}} \mathcal{M}_2$*, alors pour tout* $\sigma \in \Sigma^*$*,* $\omega \in k\text{-}\mathsf{Pile}$ *et* $q \in Q$*,*

$$(q_0,\sigma,\bot_k)\vdash^*_{\mathcal{M}_1} (q,\varepsilon,\omega) \text{ ssi } (q_0,\sigma,\bot'_\ell)\vdash^*_{\mathcal{M}_2} (q,\varepsilon,\mathcal{S}(\omega))$$

2. *Si* $\mathcal{M}_1 \leq_{\mathcal{S}} \mathcal{M}_2$*, alors le graphe* $\mathsf{Graphe}(\mathcal{M}_2)$ *(le graphe de calcul de* $\mathcal{M}_2$*) est obtenu en substituant dans* $\mathsf{Graphe}(\mathcal{M}_1)$*, chaque sommet* (p,ω) *par son image* $(p,\mathcal{S}(\omega))$ *et chaque arc issu d'une transition* δ*, par un chemin déterministe de longueur finie mais non bornée. Ce chemin correspond à la restriction du graphe de* ${\Delta_{2,\delta}}^{\vec{D}}$*, aux sommets accessibles à partir de* $(p,\mathcal{S}(\omega))$.

3. *Si* $\mathcal{F}_1 \subseteq \mathcal{F}_2$*, alors* $\mathcal{F}_1 \sqsubseteq_{\mathrm{id}} \mathcal{F}_2$*, où* id *désigne la fonction identité.*

Suite à ces remarques, les propriétés suivantes sont évidentes.

Proposition 7.1.7. *Si* $\mathcal{M}_1 \leq_{\mathcal{S}} \mathcal{M}_2$, *alors*
- $\mathrm{L}(\mathcal{M}_1) = \mathrm{L}(\mathcal{M}_2)$,
- $\mathrm{ACC}(\mathcal{M}_2) = \mathcal{S}(\mathrm{ACC}(\mathcal{M}_1))$,
- $\mathcal{M}_1$ *est déterministe ssi* $\mathcal{M}_2$ *est déterministe.*

Le résultat suivant est immédiat et sera fréquemment utilisé.

Proposition 7.1.8 (Règle de composition). *Soient* $\mathcal{F}_1$, $\mathcal{F}_2$ *et* $\mathcal{F}_3$ *des ensembles de machines. Si* $\mathcal{F}_1 \sqsubseteq_{\mathcal{S}_1} \mathcal{F}_2$ *et* $\mathcal{F}_2 \sqsubseteq_{\mathcal{S}_2} \mathcal{F}_3$, *alors* $\mathcal{F}_1 \sqsubseteq_{\mathcal{S}_1 \circ \mathcal{S}_2} \mathcal{F}_3$.

7.2 Quelques exemples

Nous illustrons maintenant cette définition en donnant quelques exemples simples de simulations déterministes.

7.2.1 Composition d'instructions versus instructions

Un des exemples les plus évidents de simulation déterministe est la simulation d'une machine à compositions d'instructions dans un ensemble $\mathfrak{I}$ par une machine à instructions dans $\mathfrak{I}$. Nous notons $\mathfrak{I}^*$ l'ensemble des instructions $\mathrm{instr} = \mathrm{instr}_1 \cdots \mathrm{instr}_n$, $n \geq 0$ désignant la composition $\mathrm{instr}_1 \circ \cdots \circ \mathrm{instr}_n$. Si $n = 0$, instr est l'instruction stay.

Par exemple, chaque élément $\mathrm{instr} = \mathrm{instr}_1 \cdots \mathrm{instr}_n \in \mathfrak{I}{instr_k}^*$, est une i-instruction, où i est le plus grand des niveaux parmi ceux de $\mathrm{instr}_1, \ldots, \mathrm{instr}_n$. Toutefois dans ce cas, le domaine de définition de instr ne dépend plus uniquement des symboles de tête des éléments. Considérons par exemple $\omega_1 = a_3[a_2[\bot]\ \bot_2]\ \bot_3$ et $\omega_2 = a_3[a_2[\bot]b_2[\bot]\ \bot_2]\ \bot_3$ et $\mathrm{instr} = \mathrm{pop}_2 \mathrm{change}_{2,a}$.

Alors $\mathrm{instr}(\omega_1)$ est indéfini, puisque $\mathrm{pop}_2(\omega_1) = a_3[\bot_2]\ \bot_3$, par contre, $\mathrm{instr}(\omega_2)$ est bien défini et égal à $a_3[a[\bot]\ \bot_2]\ \bot_3$. Pourtant, $\mathrm{top}(\omega_1) = \mathrm{top}(\omega_2)$.

Ainsi, un k-AP à instructions dans $\mathfrak{I}{instr}_k^*$ n'est simulable par un k-AP classique que si les suites d'instructions sont bien choisies. Définissons une classe d'automates à instructions composées simulable par la classe k-Pile. Il faut pour cela se restreindre aux instructions ayant un domaine de définition "identifiable". Il suffit donc d'interdire après un pop_i toute instruction qui pourrait indéfinie, c'est-à-dire, toute instruction $\mathrm{change}_{j,a}$ ou pop_j pour $j \in [1, i]$.

Définition 7.2.1. *Nous notons* ${}^*k$-AP *la classe des machines*

$$\mathcal{M} \in k\text{-MP}(\mathfrak{I}{instr}^*)$$

telles que toute transition $(p, \alpha, a_k \cdots a_1, \vec{o}, \mathrm{instr}, q)$ *de* $\mathcal{M}$ *vérifie :*

si $\mathrm{instr} = \mathrm{instr}_1 \cdots \mathrm{instr}_n$, *alors après chaque* pop_i, *le reste de la suite ne contient que des instructions* pop *et* change *de niveau strictement supérieur à* i *et des instructions* push *quelconques.*

Comparons maintenant les classes k-AP et ${}^*k$-AP.

Proposition 7.2.2. *Pour tous* $A_1, \ldots, A_k$ *et* $\vec{C}$ *contrôleur dans* k-Pile$(A_1, \ldots, A_k)$,

$$k\text{-AP}^{\vec{C}}(A_1, \ldots, A_k) \equiv_{\mathrm{id}} {}^*k\text{-AP}^{\vec{C}}(A_1, \ldots, A_k).$$

Preuve : Il est évident que $k\text{-AP}^{\vec{C}}(A_1,\ldots,A_k) \sqsubseteq_{\mathrm{id}} {}^*k\text{-AP}^{\vec{C}}(A_1,\ldots,A_k)$ puisque $k\text{-AP}^{\vec{C}}$ est inclus dans ${}^*k\text{-AP}^{\vec{C}}$.

Inversement, soit $\mathcal{A}_1 = (Q_1, \Sigma, (A_1,\ldots,A_k), \vec{C}, \Delta_1, q_0, \bot, F) \in {}^*k\text{-AP}$, nous construisons un automate $\mathcal{A}_2 = (Q_1 \cup Q_2, \Sigma, (A_1,\ldots,A_k), \vec{C}, \Delta_2, q_0, \bot, F) \in k\text{-AP}$, avec $Q_1 \cap Q_2 = \emptyset$ et Δ_2 est l'union des systèmes $\Delta_{2,\delta}$ pour $\delta \in \Delta_1$, définis de la façon suivante.

Supposons $\delta = (p, \alpha, w, \vec{o}, \mathrm{instr}, q)$ avec $\mathrm{instr} = \mathrm{instr}_1 \cdots \mathrm{instr}_n$, $n \geq 0$.

- Si $n = 0$ ou 1, alors $\Delta_{2,\delta} = \{(p, \alpha, w, \vec{o}, \mathrm{instr}, q)\}$
- si $n \geq 2$, pour que $\Delta_{2,\delta}$ soit applicable exactement aux (p,ω) pour lesquels δ est applicable, il faut être capable de déterminer par avance si $\mathrm{instr}(\delta)$ est défini. Par définition de ${}^*k\text{-AP}$, il est possible de savoir si $\mathrm{instr}(\omega)$ est défini en considérant les symboles de tête de ω.
  - Si il existe i, j tels que $\mathrm{instr}_j = \mathrm{pop}_i$ ou $\mathrm{change}_{i,a}$ et $a_i = \bot$, alors $\Delta'_\delta = \emptyset$,
  - sinon,

$$\begin{aligned}\Delta_{2,\delta} &= \{(p, \alpha, w, \vec{o}, \mathrm{instr}_1, s_{1,\delta})\} \\ &\cup \{(s_{i,\delta}, \varepsilon, w', \vec{o'}, \mathrm{instr}_{i+1}, s_{i+1,\delta}) \mid w' \in A_k^\bot \cdots A_1^\bot,\ \vec{o'} \in \{0,1\}^{|\vec{C}|}, i \in [1, n-2]\} \\ &\cup \{(s_{n-1,\delta}, \varepsilon, w', \vec{o'}, \mathrm{instr}_n, q) \mid w' \in A_k^\bot \cdots A_1^\bot,\ \vec{o'} \in \{0,1\}^{|\vec{C}|}\}.\end{aligned}$$

Clairement, tout système $\Delta_{2,\delta}$ est (p,α,q)-standard, et pour tout $\omega \in k\text{-Pile}$,

- δ est applicable à (p,ω) ssi $\Delta_{2,\delta}$ est applicable à (p,ω),
- $(p,\alpha,\omega) \vdash_{\delta^{\vec{C}}} (q,\varepsilon,\omega')$ ssi $(p,\alpha,\omega) \vdash^+_{\Delta_{2,\delta}{}^{\vec{C}}} (q,\omega')$.

L'application identité id est injective et pour tout $\delta \in \Delta$, pour tout $\omega \in k\text{-Pile}$, $\delta^{\vec{C}} \leq_{\mathrm{id},\omega} \Delta_{2,\delta}{}^{\vec{C}}$. En choisissant correctement les états appartenant à Q' (i.e., si $\delta \neq \delta'$, alors $s_{i,\delta} \neq s_{j,\delta'}$ pour tous i, j), nous obtenons $\mathcal{A}_1 \leq_{\mathrm{id}} \mathcal{A}_2$.

□

Sans la restriction sur la forme des suites d'instructions autorisées dans la définition de ${}^*k\text{-AP}$, cette construction n'aurait pas été valide puisqu'il qu'il y aurait eu dans le graphe de $\mathcal{A}_1$ des chemins interrompus par une instruction non définie et n'aboutissant donc jamais à un état de Q_1.

Pour chaque classe d'automates normalisés présentée dans la section 2.3.1, il est possible de définir une classe d'automates à suites instructions équivalente et ayant les mêmes propriétés de normalisation. Voici celle correspondant aux automates N-normalisés.

Définition 7.2.3. *Nous notons ${}^*k\text{-NAP}$ la classe des machines*

$$\mathcal{M} \in k\text{-MP}(\mathfrak{I}nstr_k{}^*)$$

telles que toute transition $(p, \alpha, a_k \cdots a_1, \vec{o}, \mathrm{instr}, q)$ de $\mathcal{M}$ vérifie :

- *si* $\mathrm{instr} = \mathrm{instr}_1 \cdots \mathrm{instr}_n$, *après chaque* pop_i, *le reste de la suite ne contient que des instructions* pop *et* change *de niveau strictement supérieur à i et des instruction* push *quelconques,*
- *si* instr *est de type* push_i, *$i < k$, alors $a_{i+1} \neq \bot$.*

Chaque pile générée par un tel automate est clairement N-normalisée.

Proposition 7.2.4. *Pour tous $A_1,\ldots,A_k$ et $\vec{C}$ contrôleur dans $k\text{-Pile}(A_1,\ldots,A_k)$,*

$$k\text{-NAP}^{\vec{C}}(A_1,\ldots,A_k) \equiv_{\mathrm{id}} {}^*k\text{-NAP}^{\vec{C}}(A_1,\ldots,A_k).$$

Voici pour terminer la classe correspondant aux automates N1-normalisés

Définition 7.2.5. *La classe* $^*k$-N1AP *des automates N1-normalisés correspond à la restriction de* $^*k$-AP *aux automates* $\mathcal{A} = (Q, \Sigma, (A_1, \ldots, A_k), \vec{C}, \Delta, q_0, \bot, F)$ *vérifiant :*

il existe une partition de Δ *telle que pour tout élément* $\overline{\delta}$ *de la partition, il existe* $p, q \in Q$, $a_1 \in A_1 \cup \{\bot\}$, $\alpha \in \Sigma \cup \{\varepsilon\}$, $\mathrm{instr} \in \mathfrak{I}nstr^*(A_1, \ldots, A_k)$ *tels que*

$$\overline{\delta} = \{(p, \alpha, a_k \cdots a_1, \vec{o}, \mathrm{instr}, q) \mid a_k \in A_k^\bot, \ldots, a_2 \in A_2^\bot\}.$$

Proposition 7.2.6. *Pour tout* $A_1, \ldots, A_k$, *pour tout contrôleur* $\vec{C}$ *dans* k-Pile$(A_1, \ldots, A_k)$,

$$k\text{-N1AP}^{\vec{C}}(A_1, \ldots, A_k) \equiv_{\mathrm{id}} {}^*k\text{-N1AP}^{\vec{C}}(A_1, \ldots, A_k).$$

Par la suite, nous nous autoriserons donc à utiliser indifféremment des automates à suites d'instructions ou à instructions simples.

7.2.2 Automates normalisés

Il est clair que pour tous $A_1, \ldots, A_k$, et contrôleur $\vec{C}$ dans k-Pile$(A_1, \ldots, A_k)$,
$k\text{-NAP}^{\vec{C}}(A_1, \ldots, A_k) \sqsubseteq_{\mathrm{id}} k\text{-AP}^{\vec{C}}(A_1, \ldots, A_k)$,
$k\text{-N2AP}^{\vec{C}}(A_1, \ldots, A_k) \sqsubseteq_{\mathrm{id}} k\text{-AP}^{\vec{C}}(A_1, \ldots, A_k)$,
$k\text{-N1AP}^{\vec{C}}(A_1, \ldots, A_k) \sqsubseteq_{\mathrm{id}} k\text{-AP}^{\vec{C}}(A_1, \ldots, A_k)$,
et si les alphabets $A_2, \ldots, A_k$ sont réduits à un unique élément ou bien si $k = 1$, alors

$$k\text{-N2AP}^{\vec{C}}(A_1, \ldots, A_k) \equiv k\text{-N1AP}^{\vec{C}}(A_1, \ldots, A_k) \equiv k\text{-AP}^{\vec{C}}(A_1, \ldots, A_k).$$

7.2.3 Automates à instructions symétriques

Soient $A_1, \ldots, A_k$ des alphabets de piles et $\vec{D} = (D_2, \ldots, D_k)$ le vecteur de contrôleurs dont chaque composante D_i contient l'ensemble des k-piles $\omega \in k$-Pile dont la projection par p_i dans i-Pile est de la forme $a[\omega_1]b[\omega_1]\omega'$.

Alors, pour tout vecteur $\vec{C}$ de contrôleurs dans k-Pile$(A_1, \ldots, A_k)$,

$$\overline{k\text{-AP}}^{\vec{C}}(A_1, \ldots, A_k) \sqsubseteq_{\mathrm{id}} k\text{-AP}^{\vec{C},\vec{D}}(A_1, \ldots, A_k).$$

En effet, chaque D_i correspond à l'union des domaines de définition des instructions $\overline{\mathrm{push}}_{i,a}$. Chaque transition $(p, \alpha, a_k \cdots a_1, \vec{o}, \overline{\mathrm{push}}_{i,a_i}, q)$ d'un automate à instructions symétriques correspond à l'union des transitions $(p, \alpha, a_k \cdots a_1, (\vec{o}, \vec{o'}), \mathrm{pop}_i, q)$ pour $\vec{o'} \in \{0,1\}^{k-1}$ dont la $i-1$-ème composante est 1.

Chapitre 8

Comparaison des classes d'automates

Nous prouvons dans ce chapitre l'équivalence entre les classes d'automates suivantes : k-AP, k-NAP, k-N1AP et k-N2AP (voir §2.3.1). Nous prendrons un soin particulier à comparer les classes de contrôleurs pour lesquelles ces équivalences sont valides. En particulier nous montrons que ces quatre classes d'automates restreintes aux contrôleurs composés d'ensembles dans PREC_k sont équivalentes.

La simulation la plus difficile à montrer est celle d'un k-AP par un k-N1AP, c'est de celle-ci que vont découler toutes les autres. Remarquons que l'équivalence entre ces deux classes d'automates à déjà été discutée en quelques lignes dans [KNU02] pour le cas des automates non contrôlés, nous montrons ici qu'elles sont équivalentes par simulation déterministe dans le cas plus général des automates contrôlés.

8.1 Simulation par un automate normalisé

Afin de prouver que k-$\mathsf{AP} \sqsubseteq k$-$\mathsf{N1AP}$, nous définissons une classe de machine intermédiaire, simulant la classe k-AP et simulable par la classe k-$\mathsf{N1AP}$. Nous définissons pour cela une injection θ de k-$\mathsf{Pile}(A_1, \ldots, A_k)$ dans k-$\mathsf{Pile}(A_1^\theta, \ldots, A_k^\theta)$ où $(A_i^\theta)_{i\geq 1}$ est une famille d'alphabets de piles construite à partir de $(A_i)_{i\geq 0}$ et dont les alphabets de niveau supérieur à 1 sont réduits à un unique élément. La classe de machines que nous allons utiliser est celle des machines dont les instructions sont les images des instructions classiques par θ.

Donnons tout d'abord une expression des alphabets A_i^θ. Soient $\#_2, \ldots, \#_k$ des nouveaux symboles n'appartenant à aucuns des alphabets A_i, nous posons :

$$A_1^\theta = \bigcup_{1\leq i\leq k} (A_i \cup \{\#_i\}) \times \cdots \times (A_1 \cup \{\#_1\}) \text{ et pour tout } i \geq 2,\ A_i^\theta = \{\#_i\}$$

Afin de manipuler plus aisément les éléments de A_1^θ, introduisons l'application

$$\# : \cup_{i\in[1,k]} A_i^\perp \cdots A_1^\perp \to A_1^\theta$$

associant à tout mot $a_i \cdots a_1$ le vecteur $\#(a_i \cdots a_1) = (a_i', \ldots, a_1')$ où $a_i' = \#_i$ si $a_i = \perp$ et $a_i' = a_i$ sinon. Cette application est bijective et toute lettre dans A_1^θ est donc l'image d'un mot dans $\mathrm{top}(i\text{-}\mathsf{Pile}(A_1, \ldots, A_i))$.

Définissons maintenant $\theta : k\text{-}\mathsf{Pile}(A_1, \ldots, A_k) \to k\text{-}\mathsf{Pile}(A_1^\theta, \ldots, A_k^\theta)$.
Pour tout $k \geq 0$, $\theta(\perp_k) = \perp_k$, et $\forall k \geq 1$, pour tout $\omega \in k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$ non vide se décomposant en $\omega = a[\omega_1]\omega'$, nous posons (avec la convention $\theta(\varepsilon) = \perp_k$),

– si $\omega_1 = \bot_{k-1}$ et $a \neq \bot$

$$\theta(\omega) = \#_k[\cdots[\#_2[\#(\mathrm{top}(\omega))\ \bot]\ \bot_2]\cdots]\,\theta(\omega')$$

– si $\omega_1 \neq \bot_{k-1}$

$$\theta(\omega) = \mathrm{change}_{1,\#(\mathrm{top}(\omega))}(\#_{k+1}[\theta(\omega_1)]\,\theta(\omega')).$$

Remarque 8.1.1.

1. *Pour tout $\omega \in k$-Pile$(A_1, \ldots, A_k)$, $\theta(\omega)$ est une pile N-normalisée (voir §1.2.1).*
2. *Par construction de θ, il est clair que pour tout $\omega \in k$-Pile :*
   – *si $\omega = \bot_k$ alors $\theta(\omega) = \bot_k$,*
   – *sinon* $\mathrm{top}(\theta(\omega)) = \#_k \cdots \#_2 \cdot \#(\mathrm{top}(\omega))$.
3. *θ est une application injective.*

Exemple 8.1.2. *Soit $\omega_{ex} = a_3[a_2[a_1a_1\ \bot]b_2[a_1\ \bot]\ \bot_2]\ \bot\ [a_2[a_1\ \bot]a_2[\bot]\ \bot_2]$ une 3-pile (voir la représentation planaire figure 8, alors $\theta_3(\omega_{ex})$ est donnée par :*

$$\#_3[\#_2[(a_3, a_2, a_1)a_1\ \bot]\#_2[(b_2, a_1)\ \bot]\ \bot_2]\#_3[\#_2[(\#_3, a_2, a_1)\ \bot]\#_2[(a_2, \#_2)\ \bot]\ \bot_2]\ \bot_3$$

puisque $\omega_{ex} = a_3[\omega_1]\ \bot\ [\omega_2]$ avec
$\theta(\omega_1) = \theta_2(a_2[a_1a_1\ \bot]b_2[a_1\ \bot]\ \bot_2) = \#_2[(a_2, a_1)a_2\ \bot]\#_2[(b_2, a_1)]\ \bot_2$ et
$\theta(\omega_2) = \theta_2(a_2[a_1\ \bot]a_2[\bot]\ \bot_2) = \#_2[(a_2, a_1)]\#_2[(a_2, \#_1)\ \bot]\ \bot_2$.

La représentation planaire de $\theta(\omega)$ est illustrée figure 19.

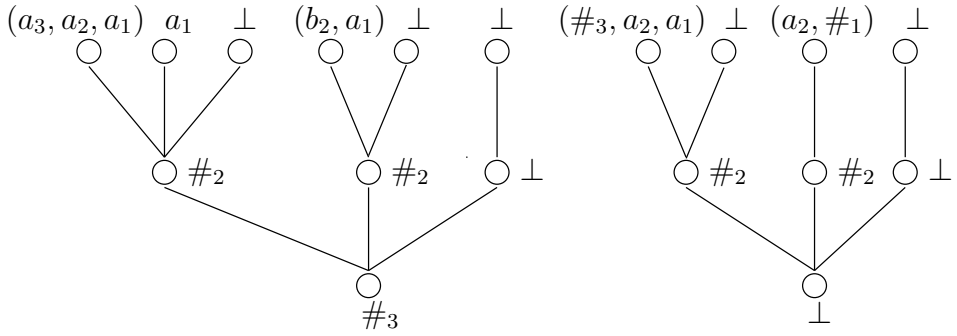


FIG. 19 – La transformation θ_3 appliquée à ω_{ex}

Afin d'obtenir une classe de machine de générant que des piles de la forme $\theta(\omega)$, nous définissons le nouvel ensemble d'instructions $\mathfrak{I}nstr^\theta$, composé de tous les instr$^\theta$ pour instr $\in \mathfrak{I}nstr$. Une instruction instr$^\theta$ envoie chaque élément $\theta(\omega)$ sur $\theta(\mathrm{instr}(\omega))$:

pour tout $\omega \in k$-Pile$(A_1^\theta, \ldots, A_k^\theta)$

– instr$^\theta(\omega) = \theta(\mathrm{instr}(\omega'))$ si il existe $\omega' \in k$-Pile$(A_1, \ldots, A_k)$ tel que $\theta(\omega') = \omega$ et instr(ω') est défini,
– instr$^\theta(\omega)$ est indéfini sinon.

La classe de machines que nous allons utiliser est donc k-MP$(\mathfrak{I}nstr^\theta)(A_1^\theta, \ldots, A_k^\theta)$. D'après la remarque 8.1.1(2), il est possible de déterminer si instr$^\theta(\theta(\omega))$ est défini, en testant les symboles de tête de $\theta(\omega)$, nous en déduisons l'équivalence suivante :.

Lemme 8.1.3. *Pour tout $k \geq 1$, pour tout $\vec{C}$ contrôleur de k-piles,*

$$k\text{-}\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k) \equiv\ k\text{-}\mathsf{MP}^{\theta(\vec{C})}(\mathfrak{I}nstr^{\theta})(A_1^{\theta}, \ldots, A_k^{\theta}).$$

Preuve : Il suffit de remarquer que pour tout instr $\in \mathfrak{I}nstr_k$, tout $a_k \cdots a_1 \in A_k^{\perp} \cdots A_1^{\perp}$, et tout $\omega \in k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$:

- $\mathrm{instr}^{\theta}(\theta(\omega)) = \theta(\mathrm{instr}(\omega))$, en particulier $\mathrm{instr}(\omega)$ est défini ssi $\mathrm{instr}^{\theta}(\theta(\omega))$ est défini,
- $\chi_{\vec{C}}(\omega) = \chi_{\theta(\vec{C})}(\theta(\omega))$,
- $\mathrm{top}(\theta(\omega)) = \#_k \cdots \#_2 \cdot \#(\mathrm{top}(\omega))$ si $\omega \neq \perp_k$, sinon $\mathrm{top}(\theta(\omega)) = \mathrm{top}(\omega)$.

Nous avons donc clairement

$$k\text{-}\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k) \sqsubseteq_{\theta}\ k\text{-}\mathsf{MP}^{\theta(\vec{C})}(\mathfrak{I}nstr^{\theta})(A_1^{\theta}, \ldots, A_k^{\theta}).$$

De la même façon, pour toute machine $\mathcal{M} \in k\text{-}\mathsf{MP}^{\theta(\vec{C})}(\mathfrak{I}nstr^{\theta})$, l'application θ^{-1} est bien définie et injective sur les piles accessibles de $\mathcal{M}$ puisqu'elles appartiennent toutes à $\theta(k\text{-}\mathsf{Pile})$. Nous avons et donc également

$$k\text{-}\mathsf{MP}^{\theta(\vec{C})}(\mathfrak{I}nstr^{\theta})(A_1^{\theta}, \ldots, A_k^{\theta}) \sqsubseteq_{\theta^{-1}} k\text{-}\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k).$$

□

D'après l'équivalence que nous venons de prouver, pour montrer que $k\text{-}\mathsf{AP} \sqsubseteq k\text{-}\mathsf{N1AP}$, il suffit de vérifier que $k\text{-}\mathsf{MP}^{\theta(\vec{C})}(\mathfrak{I}nstr^{\theta}) \sqsubseteq k\text{-}\mathsf{N1AP}$. Rappelons que dû à la structure particulière de leur système de transitions (définition 2.3.1), les transitions d'un $k\text{-}\mathsf{N1AP}$ sont données par un représentant de chaque classe d'équivalence composant le système. Tout vecteur $\delta = (p, \alpha, a_1, \mathrm{instr}, q)$, instr $\in \{\mathrm{push}_a, \mathrm{pop}_i\}$, réfère donc au système $\bar{\delta} = \{(p, \alpha, a_k \cdots a_1, \mathrm{instr}, q) \mid a_i \in A_i^{\perp}, i \in [2, k]\}$.

Afin d'éviter la multiplication des transitions, nous simulerons en fait la classe $k\text{-}\mathsf{MP}(\mathfrak{I}nstr^{\theta})$ par la classe ${}^*k\text{-}\mathsf{N1AP}$ qui est elle même équivalente à la classe $k\text{-}\mathsf{N1AP}$ d'après la proposition 7.2.6.

Lemme 8.1.4. *Pour toute transition δ d'une machine dans $k\text{-}\mathsf{MP}^{\vec{C}}(\mathfrak{I}nstr^{\theta})$, il existe un système $\bar{\Delta}_{\delta}$ d'un automate dans ${}^*k\text{-}\mathsf{N1AP}^{\vec{C}}$, tel que pour tout $\omega \in k\text{-}\mathsf{Pile}$, $\delta^{\vec{C}} \leq_{\mathrm{id},\theta(\omega)} \bar{\Delta}_{\delta}{}^{\vec{C}}$.*

Preuve : Posons

$$\delta = (p, \alpha, c_k \cdots c_1, \vec{o}, \mathrm{instr}^{\theta}, q).$$

Supposons tout d'abord que $c_k \cdots c_1 = \perp^k$. Si instr $= \mathrm{push}_{i,a}$, posons

$$\Delta_{\delta} = (p, \alpha, \perp, \vec{o}, \mathrm{push}_{\#_k} \cdots \mathrm{push}_{\#_2} \mathrm{push}_{\#(\perp^{k-i} \cdot a \cdot \perp^{i-1})}, q),$$

sinon $\Delta_{\delta} = \emptyset$.

Puisque la seule instruction définie pour $\perp_k$ est $\mathrm{push}_{i,a}^{\theta}$, alors en utilisant la remarque 8.1.1(2), et la définition de θ, il est clair que pour tout ω,

$$\delta^{\vec{C}} \leq_{\mathrm{id},\omega} \bar{\Delta}_{\delta}{}^{\vec{C}}.$$

Supposons maintenant que $c_k \cdots c_1 \neq \perp^k$, alors toujours d'après la remarque 8.1.1(2), il existe $a_k \cdots a_1$, avec $a_i \in A_i \cup \{\perp\}$ tel que

$$c_k \cdots c_1 = \#_k \cdots \#_2 \cdot \#(a_k \cdots a_1),$$

et pour tout $\omega \in k\text{-Pile}(A_1, \ldots, A_k)$,

$$\text{top}(\theta(\omega)) = c_k \cdots c_1 \text{ ssi } \text{top}(\omega) = a_k \cdots a_1. \tag{2}$$

Nous distinguons trois cas selon la valeur de instr.
Instructions de type change :
Supposons que $\text{instr} = \text{change}_a$, $a \in A_i$, posons

$$\Delta_\delta = (p, \alpha, c_1, \vec{o}, \text{change}_{\#(a_k \cdots a_{i+1} a a_{i-1} \cdots a_1)}, q).$$

Par définition de θ et (2), il est clair que pour tout $\omega \in k\text{-Pile}(A_1, \ldots, A_k)$,
$(p, \alpha, \theta(\omega)) \vdash_{\delta \vec{C}} (q, \varepsilon, \text{instr}^\theta(\omega))$ ssi $(p, \alpha, \theta(\omega)) \vdash_{\bar{\Delta_\delta} \vec{C}} (q, \varepsilon, \text{instr}^\theta(\theta(\omega)))$.
Instructions de type push :
Supposons que $\text{instr} = \text{push}_a$, $a \in A_i$. Nous posons $w_a = a_k \cdots a_{i+1} a a_{i-1} \cdots a_1$ et

$$\Delta_\delta = \{(p, \alpha, c_1, \vec{o}, \text{change}_{\#(a_i \cdots a_1)}\, \text{push}_{\#_i}\, \text{change}_{\#(w_a)}, q)\}.$$

D'après (2), pour tout $\omega \in k\text{-Pile}$ $\bar{\Delta_\delta}$ est applicable à $(p, \theta(\omega))$ ssi δ est applicable à (p, ω). Pour vérifier que $\bar{\Delta_\delta}$ satisfait le lemme, il suffit donc de prouver la propriété suivante :
$\forall \omega \in k\text{-Pile}$ tel que $\text{top}(\omega) = a_k \cdots a_1$,

$$\text{push}^\theta_{a,i}(\theta(\omega)) = \text{change}_{\#(w_a)}(\text{push}_{\#_i}(\text{change}_{\#(a_i \cdots a_1)}(\theta(\omega)))). \tag{3}$$

Vérifions cette égalité par induction sur $k \geq 1$:
Base : traitons le cas $k = 1$, dans le cas plus général où $i = k$.
Supposons que $\omega = a_k[\omega_1]\omega' \in k\text{-Pile}$,

$$\begin{aligned} \text{push}^\theta_{a,k}(\theta(\omega)) &= \text{change}_{\#(\text{top}(\text{push}_{k,a}(\omega)))}(\text{push}_{\#_k}(\theta(\omega))) \\ &= \text{change}_{\#(w_a)}(\text{push}_{\#_k}(\text{change}_{\#(\text{top}(\omega))}(\theta(\omega)))). \end{aligned}$$

En particulier, si $k = 1$, la propriété est vérifiée.
Pas d'induction : supposons la propriété vraie pour $k \geq 1$.
Soit $\omega = a_{k+1}[\omega_1]\omega' \in (k+1)\text{-Pile}$ non vide avec $\text{top}(\omega) = a_{k+1} \cdots a_1 = w$, $w' = a_k \cdots a_1$ et $a \in A_i$, $1 \leq i \leq k$. Par définition, $\text{push}_{i,a}(\omega) = a_{k+1}[\text{push}_{i,a}(\omega_1)]\omega'$.

- si $\omega_1 = \perp_k$, dans ce cas, $\theta(\omega) = \#_{k+1}[\cdots[\#_2[\#(a \perp^k) \perp] \perp] \cdots] \perp_k$, et la propriété est facile à vérifier,
- sinon,

$$\theta(\text{push}_{i,a}(\omega)) = \text{change}_{\#(a_{k+1} \cdot \text{top}(\text{push}_{i,a}(\omega_1)))}(\#_{k+1}[\theta(\text{push}_{i,a}(\omega_1))] \cdot \theta(\omega')), \tag{4}$$

et l'hypothèse d'induction appliqué à ω_1 donne :

$$\theta(\text{push}_{i,a}(\omega_1)) = \text{change}_{\#(w'_a)}(\text{push}_{\#_i}(\text{change}_{\#(a_i \cdots a_1)}(\theta(\omega_1)))). \tag{5}$$

En substituant (5) dans l'égalité (4), nous obtenons

$$\begin{aligned} \theta_{k+1}(\text{push}_{i,a}(\omega)) &= \text{change}_{\#(a_{k+1} w'_a)}(\#_{k+1}[\text{push}_{\#i}(\text{change}_{\#(a_i \cdots a_1)}(\theta(\omega_1)))]\theta(\omega')) \\ &= \text{change}_{\#(w_a)}(\text{push}_{\#_i}(\text{change}_{\#(a_i \cdots a_1)}(\theta_k(\omega)))) \\ &= \text{push}^\theta_{i,a}(\theta(\omega)). \end{aligned}$$

Instructions de type pop :

Supposons que instr $= \mathrm{pop}_i$, $1 \leq i \leq k$. Si $i = k$, alors en posant

$$\Delta_\delta = \{(p, \alpha, c_1, \vec{o}, \mathrm{pop}_k, q)\},$$

le système $\bar{\Delta}_\delta$ vérifie le lemme.

Supposons maintenant $1 \leq i < k$. Dans ce cas, posons

$$\Delta_\delta = \{(p, \alpha, c_1, \vec{o}, \mathrm{stay}, s_\delta)\} \cup \{\delta_{b_i \cdots b_1, \vec{o'}} \mid \forall j \in [1,i], b_j \in A_j \cup \{\bot\},\ \vec{o'} \in \{0,1\}^{|\vec{C}|}\},$$

où les transitions $\delta_{b_i \cdots b_1, \vec{o'}}$, sont déterminées par les valeurs de $w_1 = a_k \cdots a_{i+1}$ et $w_2 = b_i \cdots b_1$:

- si $w_2 \neq \bot^i$, alors
$$\delta_{w_2, \vec{o'}} = (s_\delta, \varepsilon, \#(w_2), \vec{o'}, \mathrm{change}_{\#(w_1 w_2)}, q),$$
- si $w_1 \neq \bot^{k-i}$ et $w_2 = \bot^i$, alors
$$\delta_{w_2, \vec{o'}} = (s_\delta, \varepsilon, \bot, \vec{o'}, \mathrm{push}_{\#_i} \cdots \mathrm{push}_{\#_2}\, \mathrm{push}_{\#(w_1 w_2)}, q),$$
- si $w_1 = \bot^{k-i}$ et $w_2 = \bot^i$,
$$\delta_{w_2, \vec{o'}} = \{(s_\delta, \varepsilon, \bot, \vec{o'}, \mathrm{pop}_k, q).$$

Nous avons clairement : $\forall \omega \in k\text{-Pile}$,

$$(p, \alpha, \theta(\omega)) \vdash_{\delta^{\vec{C}}} (q, \varepsilon, \mathrm{instr}^\theta(\omega)) \text{ ssi } (p, \alpha, \theta(\omega)) \vdash_{\Delta_\delta^{\vec{C}}} (s_\delta, \varepsilon, \mathrm{pop}_i(\theta(\omega))). \tag{6}$$

Afin de discuter de la suite du calcul du membre droit de cette équivalence, prouvons la propriété suivante par induction sur $k \geq 1$. Si $\mathrm{pop}_i(\omega)$ est défini, alors

$$\forall \omega \in k\text{-Pile}(A_1, \ldots, A_k),\ \mathrm{p}_i(\mathrm{pop}_i(\theta(\omega))) = \theta(\mathrm{pop}_i(\mathrm{p}_i(\omega))). \tag{7}$$

Base : si $i = k$, c'est évident par définition de θ. En particulier, si $k = 1$, la propriété est vraie.
Pas d'induction : supposons (7) vrai pour $k \geq 1$.

Soit $\omega \in (k+1)$-Pile et $i < k+1$, (le cas $i = k+1$ a été traité avec le cas $k = 1$) tels que $\mathrm{top}_i(\omega) \neq \bot$. Supposons $\omega = a_{k+1}[\omega_1]\omega'$, nous avons alors

$$\theta(\omega) = \mathrm{change}_{\#(\mathrm{top}(\omega))}(\#_{k+1}[\theta(\omega_1)]\theta(\omega')),$$

et

$$\begin{aligned}
\mathrm{p}_i(\mathrm{pop}_i(\theta(\omega))) &= \mathrm{p}_i(\mathrm{pop}_i(\mathrm{change}_{\#(\mathrm{top}(\omega))}(\#_{k+1}[\theta(\omega_1)]\theta(\omega')))) \text{ (par définition de } \theta) \\
&= \mathrm{p}_i(\mathrm{pop}_i(\theta(\omega_1))) \text{ (car } i \leq k) \\
&= \theta(\mathrm{p}_i(\mathrm{pop}_i(\omega_1))) \text{ (par h.r.)} \\
&= \theta(\mathrm{p}_i(\mathrm{pop}_i(\omega))) \text{ (car } i \leq k).
\end{aligned}$$

(fin de l'induction) En particulier, la propriété (7) implique directement que :

- si $\mathrm{p}_i(\mathrm{pop}_i(\omega)) \neq \bot_i$, alors $\mathrm{top}_1(\mathrm{pop}_i(\theta(\omega))) = \#(\mathrm{top}_{i \cdots 1}(\mathrm{pop}_i(\omega)))$,
- sinon $\mathrm{p}_i(\mathrm{pop}_i(\theta(\omega))) = \bot_i$.

Ainsi, pour tout ω pour lequel pop_i est défini, le système $\bar{\delta}_{\mathrm{top}_{i\cdots 1}(\mathrm{pop}_i(\omega)),\chi_{\vec{C}}(\mathrm{pop}_i(\theta(\omega)))}$ est applicable à $(s_\delta, \varepsilon, \mathrm{pop}_i(\theta(\omega)))$ et le calcul de second membre de l'équivalence (6) continue donc pour aboutir dans l'état q. Montrons que la pile obtenue à la fin de calcul est $\mathrm{pop}_i^\theta(\theta(\omega))$. Pour cela, il suffit de prouver la propriété suivante $\mathbf{P}(k)$ suivante :

pour tous $\omega \in k$-Pile , $1 \leq i < k$ tels que $\mathrm{top}_i(\omega) \neq \perp$, posons $w_1 = \mathrm{top}_{k\cdots i+1}(\omega)$ et $w_2 = \mathrm{top}_{i\cdots 1}(\mathrm{pop}_i(\omega))$,

- si $w_2 \neq \perp^i$, alors
$$\mathrm{pop}_i^\theta(\omega) = \mathrm{change}_{\#(w_1w_2)}(\mathrm{pop}_i(\omega)),$$
- si $w_1 \neq \perp^{k-i}$ et $w_2 = \perp^i$, alors
$$\mathrm{pop}_i^\theta(\omega) = \mathrm{push}_{\#(w_2w_1)}(\mathrm{push}_{\#_2} \cdots \mathrm{push}_{\#_i}(\mathrm{pop}_i(\theta(\omega)))),$$
- si $w_1 = \perp^{k-i}$ et $w_2 = \perp^i$, alors
$$\mathrm{pop}_i^\theta(\theta(\omega)) = \mathrm{pop}_k(\mathrm{pop}_i(\omega)).$$

Vérifions $\mathbf{P}(k)$ par induction sur $k \geq 2$.
Base : si $k = 2$, alors $i = 1$ et la propriété est facile à vérifier.
Pas d'induction : Supposons $\mathbf{P}(k)$ vrai pour $k \geq 2$.

Soit $\omega = a_{k+1}[\omega_1]\omega' \in (k+1)$-Pile, et $1 \leq i \leq k$ tels que $\mathrm{top}_i(\omega) \neq \perp$. Dans ce cas $\mathrm{top}_i(\omega_1) \neq \perp$. Posons

$$w_1' = \mathrm{top}_{k\cdots i+1}(\omega_1),\ w_1 = a_{k+1}(\omega)w_1' \text{ (i.e., } w_1 = \mathrm{top}_{k+1\cdots i+1}(\omega)),\ w_2 = \mathrm{top}_{i\cdots 1}(\mathrm{pop}_i(\omega)).$$

Nous distinguons trois cas selon les valeurs de w_1 et w_2.

- si $w_2 \neq \perp^i$, alors, $\mathrm{top}_{i\cdots 1}(\mathrm{pop}_i(\omega_1)) = \perp^i$ et l'hypothèse d'induction appliquée à $\mathrm{pop}_i(\omega_1)$ donne
$$\mathrm{pop}_i^\theta(\theta(\omega_1)) = \mathrm{change}_{\#(w_1'w_2)}(\mathrm{pop}_i(\theta(\omega_1)))$$
Nous avons alors pour tout $b \in A_1^\theta$
$$\mathrm{change}_b(\#_{k+1}[\theta(\mathrm{pop}_i(\omega_1))]) = \mathrm{change}_b(\#_{k+1}[\mathrm{pop}_i(\theta(\omega_1))]). \quad (8)$$
Donc
$$\begin{aligned}
\mathrm{pop}_i^\theta(\omega) &= \theta(\mathrm{pop}_i(\omega)) \text{ (par définition de } \mathrm{pop}_i^\theta) \\
&= \mathrm{change}_{\#(w_1w_2)}(\#_{k+1}[\theta(\mathrm{pop}_i(\omega_1))]\theta(\omega')) \text{ (par définition de } \theta) \\
&= \mathrm{change}_{\#(w_1w_2)}(\#_{k+1}[\mathrm{pop}_i(\theta(\omega_1))]\theta(\omega')) \text{ (par (8))} \\
&= \mathrm{change}_{\#(w_1w_2)}(\mathrm{pop}_i(\mathrm{change}_{\#(w_1w_2)}(\#_{k+1}[\theta(\omega_1)]\theta(\omega')))) \\
&= \mathrm{change}_{\#(w_1w_2)}(\mathrm{pop}_i(\theta(\omega))) \text{ (par définition de } \theta)
\end{aligned}$$
- si $w_2 = \perp^i$ et $w_1 \neq \perp^{k+1-i}$ Distinguons deux cas selon les valeurs de i et w_1' :
  - si $i < k$, et $w_1' \neq \perp^{k-i}$, dans ce cas, l'hypothèse d'induction appliquée à ω_1 donne : $\mathrm{pop}_i^\theta(\omega_1) = (\mathrm{push}_{\#(w_1'w_2)}(\mathrm{push}_{\#_2} \cdots \mathrm{push}_{\#_i}(\mathrm{pop}_i(\theta(\omega_1))))$. Nous avons donc,
$$\begin{aligned}
\mathrm{pop}_i^\theta(\theta(\omega)) &= \mathrm{change}_{\#(w_1w_2)}(\#_{k+1}[\theta(\mathrm{pop}_i(\omega_1))]\theta(\omega')) \\
\text{(par h.r.)} &= \mathrm{change}_{\#(w_1w_2)}(\mathrm{push}_{\#(w_1'w_2)}(\mathrm{push}_{\#_2} \cdots \mathrm{push}_{\#_i}(\mathrm{pop}_i(\theta(\omega))))) \\
&= \mathrm{push}_{\#(w_1w_2)))}(\mathrm{push}_{\#_2} \cdots \mathrm{push}_{\#_i}(\mathrm{pop}_i(\theta(\omega))))
\end{aligned}$$

– si $i < k$ et $w'_1 = \perp^{k-i}$, ou $i = k$, dans ce cas, $\mathrm{pop}_i(\omega_1) = \perp_k$ et donc $\mathrm{pop}_i(\omega) = a_{k+1}[\perp_k]\omega'$ et $\mathrm{pop}_i(\theta(\omega)) = \#_{k+1}[\perp_k]\theta(\omega')$.
Nous avons alors par définition de θ :

$$\begin{aligned}\mathrm{pop}_i^\theta(\theta(\omega)) &= \mathrm{push}_{\#(w_1w_2)}(\mathrm{push}_{\#_2}\cdots\mathrm{push}_{\#_i}(\#_{k+1}[\cdots\#_{i+1}[\perp_i]\cdots\perp_k]\theta(\omega')))\\ &= \mathrm{push}_{\#(w_1w_2)}(\mathrm{push}_{\#_2}\cdots\mathrm{push}_{\#_i}(\mathrm{pop}_i(\theta(\omega))))\end{aligned}$$

– si $w_2 = \perp_i$ et $w_1 = \perp^{k+1-i}$, alors clairement, $\mathrm{pop}_i(\omega) = \perp_k$ et

$$\mathrm{pop}_i^\theta(\omega) = \perp_k = \mathrm{pop}_k(\theta(\omega)) = \mathrm{pop}_k(\mathrm{pop}_i(\theta(\omega))).$$

□

Montrons maintenant que tout automate à k-piles est simulable par un automate à k-piles N1-normalisé.

Proposition 8.1.5. *Pour tout $k \geq 1$, pour tout $\vec{C}$ contrôleur de k-piles,*

$$k\text{-}\mathsf{AP}^{\vec{C}}(A_1,\ldots,A_k) \sqsubseteq_\theta k\text{-}\mathsf{N1AP}^{\theta(\vec{C})}(A_1^\theta,\ldots,A_k^\theta).$$

Preuve : D'après le lemme précédent, nous avons clairement pour tous $k \geq 0$, $\vec{C}$ vecteur de langages dans k-$\mathsf{Pile}(A_1,\ldots,A_k)$

$$k\text{-}\mathsf{MP}(\mathfrak{I}nstr_k^\theta)^{\theta(\vec{C})}(A_1^\theta,\ldots,A_k^\theta) \sqsubseteq_{\mathrm{id}} k\text{-}\mathsf{N1AP}^{\theta(\vec{C})}(A_1^\theta,\ldots,A_k^\theta).$$

En utilisant la règle de composition de la relation $\sqsubseteq$ et l'inclusion

$$k\text{-}\mathsf{AP} \sqsubseteq_\theta k\text{-}\mathsf{MP}(\mathfrak{I}nstr_k^\theta)^{\theta(\vec{C})}(A_1^\theta,\ldots,A_k^\theta)$$

donnée par le lemme 8.1.3, nous en déduisons que

$$k\text{-}\mathsf{AP}^{\vec{C}}(A_1,\ldots,A_k) \sqsubseteq_\theta k\text{-}\mathsf{N1AP}^{\theta(\vec{C})}(A_1^\theta,\ldots,A_k^\theta).$$

□

Nous pouvons remarquer de plus que l'automate que nous venons de construire est N-normalisé (il l'est forcément car d'après la remarque 8.1.1(1), il ne génère que des piles N-normalisées), et N2-normalisé puisque l'instruction change n'est jamais utilisée aux niveaux supérieurs à 1 (et une instruction $\mathrm{change}_{1,a}$ correspond à $\mathrm{pop}_1\,\mathrm{push}_{1,a}$). Nous avons donc également :

Proposition 8.1.6. *Pour tout $k \geq 1$, pour tout $\vec{C}$ contrôleur dans k-$\mathsf{Pile}(A_1,\ldots,A_k)$,*
$k\text{-}\mathsf{AP}^{\vec{C}}(A_1,\ldots,A_k) \sqsubseteq_\theta$

$$k\text{-}\mathsf{NAP}^{\theta(\vec{C})}(A_1^\theta,\ldots,A_k^\theta) \cap k\text{-}\mathsf{N2AP}^{\theta(\vec{C})}(A_1^\theta,\ldots,A_k^\theta) \cap k\text{-}\mathsf{N1AP}^{\theta(\vec{C})}(A_1^\theta,\ldots,A_k^\theta).$$

8.2 Équivalences

Nous regardons maintenant comment les simulations définies dans la section précédente modifient les classes de contrôleurs, lorsque ces derniers sont des ensembles SOM-définissables (c'est à dire, des éléments de PREC_k). Nous avons pour cela besoin du résultat suivant qui est un corollaire immédiat de la proposition 8.1.5.

Lemme 8.2.1. *Pour tout $k \geq 1$, pour tout $\vec{C}$ contrôleur de k-piles sur les alphabets $A_1, \ldots, A_k$,*

$$k\text{-}\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k) \sqsubseteq_\theta k\text{-}\mathsf{AP}^{\theta(\vec{C})}(A_1^\theta, \ldots, A_k^\theta).$$

Preuve : Nous avons remarqué lors des premiers exemples de simulations (voir §7.2.2) que pour tous alphabets de piles $B_1, \ldots, B_k$, pour tout $\vec{D}$ vecteur d'ensembles dans $k\text{-}\mathsf{Pile}(B_1, \ldots, B_k)$,

$$k\text{-}\mathsf{N1AP}^{\vec{D}}(B_1, \ldots, B_k) \sqsubseteq_{\mathrm{id}} k\text{-}\mathsf{AP}^{\vec{D}}(B_1, \ldots, B_k),$$

en composant ce résultat avec la proposition 8.1.5, le corollaire est démontré.
□

Proposition 8.2.2. *Pour tout $D \subseteq k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$,*
si $D \in \mathsf{PREC}_k(A_1, \ldots, A_k)$, alors $\theta(D)$ appartient à $\mathsf{PREC}_k(A_1^\theta, \ldots, A_k^\theta)$.

Preuve : Démontrons la proposition par induction sur $k \geq 1$.
Base : si $k = 1$, alors θ est la fonction identité et D appartiennent donc bien à $\mathsf{PREC}_1(A_1^\theta)$.
Pas d'induction : supposons la propriété vraie pour $k \geq 1$. Soit $D \in \mathsf{PREC}_{k+1}(A_1, \ldots, A_{k+1})$. D'après le théorème 6.2.7, il existe un vecteur $\vec{C} = (C_1, \ldots, C_m)$, dont chaque composante est de la forme $C_i = {\mathrm{p}_k}^{-1}(R_i)$, où R_i est un élément de $\mathsf{PREC}_k(A_1, \ldots, A_k)$, et tel que $D \in \mathsf{ACC}_{k+1}^{\vec{C}}$. D'après le lemme 8.2.1 et les propriétés de la simulation déterministe (voir proposition 7.1.7), $\theta(D)$ appartient donc à $\mathsf{ACC}_k^{\theta(\vec{C})}(A_1^\theta, \ldots, A_k^\theta)$. Alors, en appliquant encore le théorème 6.2.7, si les composantes de $\theta(\vec{C})$ appartiennent à PREC_{k+1}, alors $\theta(D)$ aussi. Chaque $\theta(C_i) = \theta({\mathrm{p}_k}^{-1}(R_i))$ s'écrit de la façon suivante :

$$\theta((k+1)\text{-}\mathsf{Pile}) \cap \bigcup_{a_{k+1}\cdots a_1} \{\omega \mid \mathrm{top}_1(\omega) = \#(a_{k+1}\cdots a_1),\ \mathrm{change}_{\#(a_k\cdots a_1)}(\omega) \in {\mathrm{p}_k}^{-1}(\theta(R_i))\}.$$

Or,

1. par hypothèse de récurrence, $\theta(R_i)$ appartient à $\mathsf{PREC}_k(A_1^\theta, \ldots, A_k^\theta)$ et en appliquant la proposition 6.2.5, ${\mathrm{p}_k}^{-1}(\theta(R_i))$ appartient à $\mathsf{PREC}_{k+1}(A_1^\theta, \ldots, A_{k+1}^\theta)$,
2. $\theta((k+1)\text{-}\mathsf{Pile})$ appartient à $\mathsf{PREC}_{k+1}(A_1^\theta, \ldots, A_{k+1}^\theta)$, car il correspond au langage de pile généré par l'automate obtenu en appliquant le lemme 8.2.1 à un k-AP non contrôlé générant $(k+1)$-Pile.

La classe des PREC_{k+1} étant close par intersection (car égale à DEF_k), nous en déduisons que chaque $\theta(C_i)$ est définissable en LSOM dans $\mathbf{Pile_{k+1}}$ et appartient donc à PREC_{k+1}. Ainsi, $\theta(D)$ est généré par un système à $(k+1)$-piles contrôlés par des éléments de PREC_{k+1}, et par suite, $\theta(D) \in \mathsf{PREC}_{k+1}(A_1^\theta, \ldots, A_{k+1}^\theta)$.

La propriété est donc vraie pour tout $k \geq 1$, ce qui démontre la proposition.
□

Corollaire 8.2.3. *Pour tous alphabets de piles $A_1, \ldots, A_k$, $k \geq 0$, il existe des alphabets de piles $B_1, \ldots, B_k$ tels que,*

pour tout vecteur $\vec{C}$ contrôleurs dans i-$\mathsf{Pile}(A_1, \ldots, A_i)$, il existe un vecteur $\vec{D}$ de contrôleurs dans i-$\mathsf{Pile}(B_1, \ldots, B_i)$, tel que

$$k\text{-}\mathsf{AP}^{[\vec{C}]_i}(A_1, \ldots, A_k) \sqsubseteq k\text{-}\mathsf{N1AP}^{[\vec{D}]_i}(B_1, \ldots, B_k)$$

et

- *$\vec{D}$ et $\vec{C}$ sont de même taille,*
- *si $\vec{C} \in \mathsf{PREC}_i(A_1, \ldots, A_i)$, alors $\vec{D} \in \mathsf{PREC}_i(B_1, \ldots, B_i)$*
- *$B_2, \ldots, B_k$ sont réduits à un unique élément.*

La notation $[\vec{C}]_i$ désigne le vecteur $[\vec{C}]_{k,i}$ défini dans l'introduction de cette partie : $[\vec{C}]_{k,i}$ est composé de tous les $\{\omega \in k\text{-}\mathsf{Pile} \mid \mathrm{p}_i(\omega) \in C_j\}$. Nous nous autorisons à supprimer l'indice k puisqu'il est donné par le niveau de la classe d'automates. Nous nous autoriserons cet allègement de notation dès qu'il sera possible.

Preuve : Le corollaire est démontré pour la cas $i = k$, puisque d'après la proposition 8.1.5,

$$k\text{-}\mathsf{AP}^{[\vec{C}]_k}(A_1, \ldots, A_k) \sqsubseteq k\text{-}\mathsf{N1AP}^{[\theta(\vec{C})]_k}(A_1^\theta, \ldots, A_k^\theta)$$

et d'après la proposition 8.2.2, $\theta(\vec{C})$ est composé d'éléments de PREC_k.

Supposons maintenant que $i \in [0, k]$, et $\vec{C} = (C_1, \ldots, C_m)$ où chaque composante C_j appartient à PREC_i. $\theta([\vec{C}]_{k,i}) = [\mathrm{p}_i(\theta([\vec{C}]_i))]_{k,i}$, puisque pour tous $\omega \in k\text{-}\mathsf{Pile}$, $j \in [1, m]$,

$$\mathrm{p}_i(\omega) \in C_j \text{ ssi } \mathrm{p}_i(\theta(\omega)) \in \mathrm{p}_i(\theta([C_j]_{i,k})).$$

En appliquant la proposition 8.1.5, nous obtenons donc

$$k\text{-}\mathsf{AP}^{[\vec{C}]_i}(A_1, \ldots, A_k) \sqsubseteq k\text{-}\mathsf{N1AP}^{[\vec{D}]_i}(A_1^\theta, \ldots, A_k^\theta)$$

où $\vec{D} = (D_1, \ldots, D_m)$ et pour tout $j \in [1, m]$, $D_j = \mathrm{p}_i(\theta([C_j]_{i,k}))$. Puisque les C_j appartiennent à PREC_i, d'après la proposition 6.2.5, $[\vec{C_j}]_{i,k}$ appartient à PREC_k, et donc d'après le lemme 8.2.2, $\theta([\vec{C_j}]_{i,k}) \in \mathsf{PREC}_k$. Enfin, en utilisant la proposition 6.2.5, nous obtenons $D_j \in \mathsf{PREC}_i$, ce qui achève la preuve.

□

Pour conclure, nous avons remarqué au §7.2.2 que sur des alphabets de piles réduits à un élément à partir du niveau 2, les classes k-$\mathsf{AP}^{\vec{C}}$, k-$\mathsf{N1AP}^{\vec{C}}$, k-$\mathsf{N2AP}^{\vec{C}}$, étaient équivalentes. Nous en déduisons le théorème général suivant :

Théorème 8.2.4. *Pour tout $k \geq 1$,*

$$k\text{-}\mathsf{AP}^{\mathsf{PREC}_k} \equiv k\text{-}\mathsf{N2AP}^{\mathsf{PREC}_k} \equiv k\text{-}\mathsf{NAP}^{\mathsf{PREC}_k} \equiv k\text{-}\mathsf{N1AP}^{\mathsf{PREC}_k}$$

et

$$k\text{-}\mathsf{AP}^{\emptyset} \equiv k\text{-}\mathsf{N2AP}^{\emptyset} \equiv k\text{-}\mathsf{NAP}^{\emptyset} \equiv k\text{-}\mathsf{N1AP}^{\emptyset}.$$

Chapitre 9

Simulation des contrôleurs

Nous définissons dans cette section une restriction de k-Pile permettant de coder, plus finement que par la bijection ${\varphi_k}^{-1}$, les éléments de $\mathcal{P}_k$. Nous appelons ces piles des $\mathcal{P}_k$-piles. Nous utilisons une classe de machines, appelées $\mathcal{P}_k$-AP, ne générant que des $\mathcal{P}_k$-piles, par l'application d'instructions envoyant chaque $\mathcal{P}_k$-pile codant un mot u sur la $\mathcal{P}_k$-pile codant $u \bullet a$.

Nous définissons dans le première section, le codage $\mathcal{C}_k$, ainsi que les nouvelles instructions permettant de travailler sur les éléments $\mathcal{C}_k(u)$ (les $\mathcal{P}_k$-piles) et une classe de machines utilisant ces instructions. Dans la deuxième section nous étudions la simulation d'un k-AP par un $\mathcal{P}_k$-AP, puis dans la troisième section, nous exploitons les propriétés du codage pour montrer que les automates à $\mathcal{P}_k$-piles sont simulables par différents types de k-AP. Enfin, nous traduisons les résultats obtenus dans la section 3 en terme d'automates à k-piles. Nous concluons principalement : à l'équivalence (par simulation déterministe) des k-AP non contrôlés

- avec les automates à k-piles symétriques non contrôlé,
- avec les automates à k-piles contrôlés par des éléments de PREC_k,
- avec les automates à 1-piles contrôlées par un langage de niveau k déterministe.

9.1 Automates à $\mathcal{P}_k$-piles

9.1.1 Codage des mots de piles

Nous disposons d'une bijection entre k-Pile et un langage $\mathcal{P}_k$ clos par préfixe qui nous a permis d'étudier la logique sur les k-piles (voir §1.4 pour un rappel des définitions). Nous exploitons à nouveau cette représentation des k-piles, afin d'étudier les éventuelles équivalences entre automates à k-piles et automates à k-piles contrôlés. Nous définissons pour cela un nouveau codage des éléments $u \in \mathcal{P}_k$ sous forme de k-piles, permettant un accès direct à chacune des projections $\mathrm{f}_i(u)$ de u dans $\mathcal{P}_i$.

Nous définissons pour chaque lettre a dans l'alphabet $\widehat{A_{1,k}}$, une nouvelle instruction de pile $\mathrm{prod}_{k,a}$, envoyant tout $\mathcal{C}_k(u)$ tel que $u \bullet a \in \mathcal{P}_k$, sur $\mathcal{C}_k(u \bullet a)$. Nous utilisons ensuite ces nouvelles instructions pour définir de nouvelles machines ayant la particularité de ne générer que des k-piles de la forme $\mathcal{C}_k(u)$ pour $u \in k$-Pile.

Posons tout d'abord quelques notations. Soient $A_1, \ldots, A_k, \ldots$ des alphabets de piles fixés pour toute la suite. Nous désignerons par $A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}}, \ldots$ les alphabets définis par $A_1^{\mathcal{C}} = A_1$ et

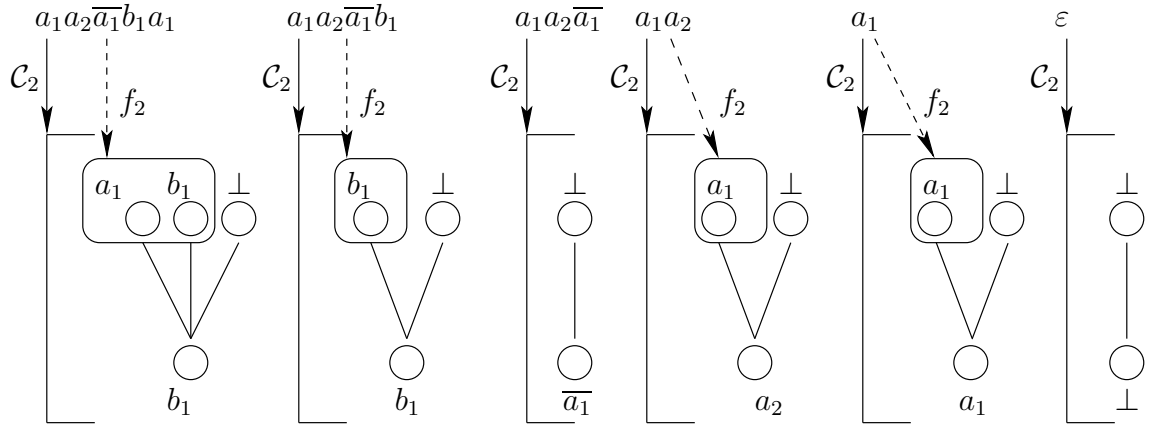


FIG. 20 – construction de $\mathcal{C}_2(a_1a_2\overline{a_1}b_1a_1)$

pour tout $i \geq 1$, $A^{\mathcal{C}}_{i+1} = A_{i+1} \cup \widehat{A_{1,i}}$. Remarquons que $A^{\mathcal{C}}_1 \subseteq A^{\mathcal{C}}_2 \ldots \subseteq A^{\mathcal{C}}_k \ldots$

Les éléments de $u \in \mathcal{P}_k(A_1, \ldots, A_k)$ seront envoyés injectivement par $\mathcal{C}_k$ dans l'ensemble des k-piles sur les alphabets de piles $A^{\mathcal{C}}_1, \ldots, A^{\mathcal{C}}_k$, ou plus rigoureusement, sur des copies de chacun de ces alphabets, ceci afin d'avoir des alphabets disjoints. Afin d'éviter la multiplication des indices, nous omettrons (abusivement) de distinguer les différentes copies et travaillerons directement sur les ensembles $A^{\mathcal{C}}_1, \ldots, A^{\mathcal{C}}_k$.

Définition 9.1.1. *Pour tout $k \geq 0$, l'application $\mathcal{C}_k : \mathcal{P}_k(A_1, \ldots, A_k) \to k\text{-}\mathsf{Pile}(A^{\mathcal{C}}_1, \ldots, A^{\mathcal{C}}_k)$ est définie inductivement de la façon suivante*

- $\mathcal{C}_k(\varepsilon) = \perp_k$,
- *pour tout $u \cdot a \in \mathcal{P}_{k+1}$,*

$$\mathcal{C}_{k+1}(u \cdot a) = a[\mathcal{C}_k(\mathrm{f}_k(u \cdot a))] \cdot \mathcal{C}_{k+1}(u)$$

Nous notons $\mathcal{P}_k$-Pile l'ensemble des images par $\mathcal{C}_k$ des éléments de $\mathcal{P}_k$.

Par définition, si $u = a_1 \cdots a_n \in \mathcal{P}_{k+1}$, alors, en posant $u_i = a_1 \cdots a_i$ pour tout $i \in [1, n]$,

$$\mathcal{C}_{k+1}(u_n) = a_n[\mathcal{C}_k(\mathrm{f}_k(u_n))]a_{n-1}[\mathcal{C}_k(\mathrm{f}_k(u_{n-1}))] \cdots a_1[\mathcal{C}_k(\mathrm{f}_k(u_1))]\ \perp_{k+1}$$

Exemple 9.1.2. *Soit $A_1 = \{a_1, b_1\}$, $A_2 = \{a_2\}$ et $u = a_1a_2\bar{a}_1b_1b_1 \in \mathcal{P}_2$ (qui correspond à la pile $\varphi_2{}^{-1}(u) = a_2[b_1b_1\ \perp]\ \perp\ [a_1\ \perp]$).*

Alors $\mathcal{C}_2(u) = b_1[b_1b_1\ \perp]b_1[b_1\ \perp]\bar{a}_1[\perp]a_2[a_1\ \perp]a_1[a_1\ \perp]\ \perp_2$.

La figure 9.1.1 détaille les étapes de la construction de $\mathcal{C}_2(u)$.

Remarque 9.1.3.

1. *Par définition, le mot de tête de niveau k de $\mathcal{C}_k(u)$ est u, et pour tout $i < k$, le mot de tête de niveau i de $\mathcal{C}_k(u)$ est la projection de u dans $\mathcal{P}_i$.*

   Formellement, nous avons pour tout $1 \leq i \leq k$:

   $\mathrm{mot}_i(\mathcal{C}_k(u)) = \mathrm{f}_{k,i}(u)$. *De plus,* $\mathrm{p}_i(\mathcal{C}_k(u)) = \mathcal{C}_i(\mathrm{f}_{k,i}(u))$ *et est donc complètement déterminé par* $\mathrm{mot}_i(\mathcal{C}_k(u))$.

2. *pour tout $k \geq 0$, $\mathcal{C}_k$ est une injection.*

Comme pour θ précédemment, nous définissons des instructions laissant stable les $\mathcal{P}_k$-piles et permettant de générer $\mathcal{P}_k$-Pile.

Définition 9.1.4. *Pour tout $i \geq 1$ et $a \in A_i \cup \overline{A_i}$, l'instruction* prod_a *est définie pour tout $\omega \in k\text{-Pile}(A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}})$, $k \geq 0$, par*

- $\mathrm{prod}_{k,a}(\omega) = \mathcal{C}_k(u \bullet a)$ *si $i \in [1,k]$, et il existe $u \in \mathcal{P}_k$ tel que $\mathcal{C}_k(u) = w$] et $u \bullet a \in \mathcal{P}_k$,*
- $\mathrm{prod}_{k,a}(\omega)$ *est indéfini sinon.*

Lemme 9.1.5. *Pour tous $u \in \mathcal{P}_k$, et $a \in A_i \cup \overline{A_i}$, $1 \leq i \leq k$,*

- *si $a \in A_i$, alors* $\mathrm{prod}_a(\mathcal{C}_k(u))$ *est défini*
- *si $a \in \overline{A_i}$, alors* $\mathrm{prod}_a(\mathcal{C}_k(u))$ *est défini ssi* $\mathrm{top}_i(\mathcal{C}_k(u)) = \overline{a}$

Preuve : D'après la proposition 1.4.4,

- si $a \in A_i$, alors $u \bullet a \in \mathcal{P}_k$. Dans ce cas $\mathrm{prod}_{k,a}(\mathcal{C}(u))$ est donc défini
- sinon, $u \bullet a \in \mathcal{P}_k$ ssi $\mathrm{fin}(\mathrm{f}_i(u)) = \overline{a}$. D'après la remarque 9.1.3, $\mathrm{fin}(\mathrm{f}_i(u)) = \mathrm{top}_i(\mathcal{C}_k(u))$ et donc $\mathrm{prod}_{k,a}(\mathcal{C}(u))$ est défini ssi $\mathrm{top}_i(\mathcal{C}_k(u)) = \overline{a}$.

□

Définissons maintenant une classe de machine à instructions dans $\mathfrak{I}nstr_{\mathcal{P}_k} = \{\mathrm{prod}_{k,a} \mid a \in \widehat{A_{1,k}}\}$, et ne générant que des piles de la forme $\mathcal{C}_k(u)$.

Définition 9.1.6. *Soient $k \geq 1$ et $A_1, \ldots, A_k$ des alphabets de piles.*
Nous notons $\mathcal{P}_k$-AP$(A_1, \ldots, A_k)$ l'ensemble des machines

$$\mathcal{M} \in k\text{-MP}(\mathfrak{I}nstr_{\mathcal{P}_k}, (A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}}))$$

telles que toute transition $(q, \alpha, a_k \ldots a_1, \vec{o}, \mathrm{prod}_{k,a}, p)$ vérifie :
si $a \in \overline{A_i}$, $1 \leq i \leq k$, alors $a_i = \overline{a}$

Remarque 9.1.7.

1. *Soit $\delta = (q, \alpha, a_k \ldots a_1, \vec{o}, \mathrm{prod}_{k,a}, p)$ une transition d'un $\mathcal{P}_k$-AP, alors pour tout $u \in \mathcal{P}_k$, $\mathrm{top}(\mathcal{C}_k(u)) = a_k \cdots a_1$ ssi $\mathrm{prod}_{k,a}(\omega)$ est défini.*
2. *L'ensemble des piles générées par un automate à $\mathcal{P}_k$-piles est inclus dans $\mathcal{P}_k$-Pile.*
3. *Pour tout vecteur $\vec{R}$ de langages dans $\mathcal{P}_k$,*

$$\mathcal{P}_k\text{-AP}^{(\vec{R})_k} \equiv_{\mathrm{id}} \mathcal{P}_k\text{-AP}^{\mathcal{C}_k(\vec{R})}$$

puisque pour tout $u \in \mathcal{P}_k$, $\mathrm{mot}_k(\mathcal{C}_k(u)) \in R_j$ ssi $u \in R_j$. (Nous nous autorisons la même simplification que pour $[R]_{k,i}$: ici $(\vec{R})_k$ désigne en fait le controleur $(\vec{R})_{k,k}$. Rappelons que $(R)_{k,i} = \{\omega \in k\text{-Pile} \mid \mathrm{mot}_i(\omega) \in R$ que nous noterons lorsqu'il n'y a pas de confusion possible, i.e., lorsque $(R)_i$ contrôle un automate de niveau k.)
De la même façon, pour tout vecteur $\vec{R}$ de langages dans $\mathcal{P}_i$,

$$\mathcal{P}_k\text{-AP}^{(\vec{R})_i} \equiv_{\mathrm{id}} \mathcal{P}_k\text{-AP}^{[\mathcal{C}_k(\vec{R})]_i}$$

4. *Nous notons $\mathcal{P}_k$-AP$^{(\mathsf{REC}_i)_i}$ l'ensemble des $\mathcal{P}_k$-AP$^{(\vec{R})_i}$, pour $\vec{R}$ vecteur de langages dans REC_i*

9.1.2 Exemples

Nous terminons cette section par trois exemples de langages reconnus par ces machines.

Proposition 9.1.8. *Tout langage dans* REC_{k+1}, $k \geq 0$ *est reconnu par un automate à* $\mathcal{P}_k$*-pile déterministe contrôlé par un vecteur de langages dans* REC_k.

Preuve : D'après le théorème 6.2.7, tout langage dans REC_{k+1} est reconnu par un automate fini à oracle $\mathcal{A} = (Q, f_k^{-1}(\vec{R}), \Delta, p_0, F) \in \mathsf{AF}_{k+1}(A_1, \ldots, A_{k+1})$ où $\vec{R}$ est composé de langages réguliers. D'après la proposition 4.1.8, nous pouvons, sans perte de généralité, supposer que $\mathcal{A}$ est déterministe et complet dans $\mathcal{P}_{k+1}$.

Nous associons à $\mathcal{A}$, l'automate $\mathcal{A} = (Q, A_{k+1}^{\mathcal{C}}, (A_1^{\mathcal{C}}, \ldots A_k^{\mathcal{C}}), (\vec{R})_k, \Delta_{\mathcal{A}}, p_0, F) \in \mathcal{P}_k$-$\mathsf{AP}$, où $\Delta_{\mathcal{A}}$ est obtenu en posant pour toute transition

$$(q, a, \vec{o}, q') \in \Delta$$

- si $a \in A_{k+1}$, $\forall a_k \ldots a_1$, $a_i \in A_i^{\mathcal{C}} \cup \{\bot\}$ alors

$$(q, a, a_k \cdots a_1, \vec{o}, \text{stay}, q') \in \Delta_{\mathcal{A}}$$

- si $a \in \mathcal{A}_k$, alors $\forall a_k \cdots a_1$, $a_i \in A_i^{\mathcal{C}} \cup \{\bot\}$, alors

$$(q, a, a_k \cdots a_1, \vec{o}, \text{prod}_{a,k}, q') \in \Delta_{\mathcal{A}}$$

- si $a \in \overline{A_i}$, $i \in [1, k]$, alors pour tous $a_k \cdots a_1$, tels que $a_j \in A_j^{\mathcal{C}} \cup \{\bot\}$, et $a_i = \overline{a}$, alors

$$(q, a, a_k \cdots a_1, \vec{o}, \text{prod}_{a,k}, q') \in \Delta_{\mathcal{A}}$$

Puisque $\mathcal{A}$ est déterministe, $\mathcal{A}$ est également déterministe. L'automate $\mathcal{A}$ étant complet dans $\mathcal{P}_{k+1}$, définissons $\eta : \mathcal{P}_{k+1} \to Q$ associant à tout u l'unique p tel que $(p_{0,\uparrow}\, u) \to_{\mathcal{A}}^* (p, u_\uparrow)$.

Une induction évidente sur $n \geq 0$ permet de montrer que pour tout $u \in A_{k+1}^{\mathcal{C}}$ et $\omega \in k$-Pile,

$$(p_0, u, \bot_k) \vdash_n^{\mathcal{A}} (p, \varepsilon, \omega) \text{ ssi } u \in \mathcal{P}_{k+1},\ \omega = \mathcal{C}_k(\mathrm{f}_k(u))) \text{ et } p = \eta(u)$$

L'automate $\mathcal{A}$ reconnaît donc $\mathrm{L}(\mathcal{A})$.

□

Il est en fait possible de montrer une propriété plus forte : tout vecteur $\vec{R}$ composé de langages dans REC_{k+1} est reconnu par un automate à $\mathcal{P}_k$-piles déterministe. Nous entendons par là qu'avec le même $\mathcal{P}_k$-AP déterministe, il est possible de reconnaître chacune des composantes de $\vec{R}$, par un choix adéquat des ensembles d'états finaux.

Proposition 9.1.9. *Pour tout vecteur* $\vec{L} = (L_1, \ldots, L_m)$ *de langages dans* REC_{k+1}, *il existe* m *automates déterministes*

$$\mathcal{A}_i = (Q, \Sigma, (A_1, \ldots, A_k), (\vec{R})_k, \Delta, q_0, \bot, F_i) \in \mathcal{P}_k\text{-}\mathsf{AP}^{(\mathsf{REC}_k)_k},\ \textit{pour } i \in [1, m]$$

tels que pour tout $i \in [1, m]$, $\mathrm{L}(\mathcal{A}_i) = L_i$.

Preuve : Supposons pour simplifier que $\vec{R} = (R_1, R_2)$, avec $R_i \in \mathsf{REC}_{k+1}$ et chaque R_i est reconnu par un automate déterministe et complet $\mathcal{A}_i \in \mathsf{AF}_{k+1}$ à oracles ${\mathrm{f}_k}^{-1}(\vec{O^i})$. Sans perte de généralité, nous pouvons supposer que $\vec{O^1} = \vec{O^2} = \vec{O}$. (Sinon, on peut trouver de

nouveaux automates à oracles $(\mathrm{f}_k{}^{-1}(\vec{O_1}), \mathrm{f}_k{}^{-1}(\vec{O_2}))$ et reconnaissant les mêmes langages.) Par produit des automates $\mathcal{A}_1$ et $\mathcal{A}_2$, nous obtenons un nouvel automate déterministe et complet $\mathcal{A} = (Q_1 \times Q_2, \delta, (p_{0,1}, p_{0,2}), F_1 \times F_2) \in \mathsf{AF}_{k+1}$, contrôle par $\mathrm{f}_k{}^{-1}(\vec{O})$, et vérifiant pour tout $u \in \mathcal{P}_{k+1}$:

$$\text{si } ((p_{0,1}, p_{0,2}), {}_{\uparrow}u) \to ((p_1, p_2), u_{\uparrow}) \text{ alors } \forall i \in \{1,2\},\ u \in R_i \text{ ssi } p_i \in F_i$$

En appliquant la preuve précédente à cette automate nous construisons pour tout $i \in \{1,2\}$, $\mathcal{A}_i = (Q_1 \times Q_2, A^{\mathcal{C}}_{k+1}, (A^{\mathcal{C}}_1, \dots A^{\mathcal{C}}_k), (\vec{R})_k, \Delta_{\mathcal{A}}, (p_{0,1}, p_{0,2}), F'_i) \in \mathcal{P}_k\text{-}\mathsf{AP}$, où $F'_1 = F_1 \times Q_2$ et $F'_2 = Q_1 \times F_2$.

□

Le langage des mouvements de piles $\mathcal{M}_k$ (défini dans le chapitre §1.4) est reconnu par un $\mathcal{P}_k$-AP non contrôlé.

Lemme 9.1.10. *Pour tout $k \geq 0$, le langage $\mathcal{M}_k(A_1, \dots, A_k)$ est reconnu par un automate dans $\mathcal{P}_k$-$\mathsf{AP}(A_1, \dots, A_k)$.*

Preuve : Posons $\mathcal{A} = (\{q_0\}, \widehat{A_{1,k}}, (A^{\mathcal{C}}_1, \dots, A^{\mathcal{C}}_k), \Delta, q_0, \bot, \{q_0\})$ et

$$\Delta = \{(q_0, a, a_k \cdots a_1, \mathrm{prod}_a, q_0) \mid a_i \in A^{\mathcal{C}}_i \cup \{\bot\}\}$$

Par définition, pour tout $u \in \mathcal{M}_k$ et $a \in \widehat{A_{1,k}}$, $ua \in \mathcal{M}_k$ ssi $\rho(u) \bullet a \in \mathcal{P}_k$. Les seuls mots admettant un calcul dans $\mathcal{A}_k$ sont donc les éléments de $\mathcal{M}_k$, et $\forall u \in \mathcal{M}_k$,

$$(q_0, u, \bot_k) \vdash^*_{\mathcal{A}} (q, \varepsilon, \mathcal{C}_k(\rho(u)))$$

□

Exemple 9.1.11 ($\mathcal{P}_k$-AP deterministe reconnaissant $\mathcal{M}_k$). *Voici un calcul du mot $u = c_1\bar{c_1}b_1a_2a_1a_3\bar{a_1}\bar{a}_2 \in \mathcal{M}_3$ pour $A_1 = \{a_1, b_1, c_1\}$, $A_i = \{a_i\}$, $i \in \{2,3\}$*

$$\begin{aligned}
(q_0, u, \bot_3) &\vdash (q_0, \bar{c_1}b_1a_2a_1a_3\bar{a_1}a_2, \omega_1) \textit{ avec } \omega_1 = c_1[c_1[c_1\ \bot]\ \bot\ [\bot]]\ \bot_3 \\
&\vdash (q_0, b_1a_2a_1a_3\bar{a_1}\bar{a}_2, \bot_3) \\
&\vdash (q_0, a_1a_3\bar{a_1}\bar{a_2}, \omega_2) \textit{ avec } \omega_2 = b_1[b_1[b_1\ \bot]\ \bot\ [\bot]]\ \bot_3) \vdash^{a_2} \\
&\vdash (q_0, a_2a_1a_3\bar{a_1}\bar{a_2}, \omega_3) \textit{ avec } \omega_3 = a_2[a_2[b_1\ \bot]b_1[b_1\ \bot]\ \bot\ [\bot]]\omega_2 \\
&\vdash (q_0, a_3\bar{a_1}\bar{a_2}, \omega_4) \textit{ avec } \omega_4 = a_1[a_1[a_1b_1\ \bot]a_2[b_1\ \bot]b_1[b_1\ \bot]\ \bot\ [\bot]]\omega_3 \\
&\vdash (q_0, \bar{a_1}\bar{a_2}, \omega_5) \textit{ avec } \omega_5 = a_3[a_1[a_1b_1\ \bot]a_2[b_1\ \bot]b_1[b_1\ \bot]\ \bot\ [\bot]]\omega_4 \\
&\vdash (q_0, \bar{a_2}, \omega_6) \textit{ avec } \omega_6 = \bar{a_1}[a_2[b_1\ \bot]b_1[b_1\ \bot]\ \bot\ [\bot]]\omega_4 \\
&\vdash (q_0, \varepsilon, \omega_7) \textit{ avec } \omega_7 = \bar{a_2}[b_1[b_1\ \bot]\ \bot\ [\bot]]\omega_4)
\end{aligned}$$

Nous avons alors bien $\omega_7 = \mathcal{C}_3(\rho(u))$ car
$\rho(u) = b_1a_2a_1a_3\bar{a_1}\bar{a_2} = \mathrm{mot}_3(\omega_7)$, $\mathrm{f}_2(\rho(u)) = b = \mathrm{mot}_2(\omega_7)$ *et* $\mathrm{f}_1(\rho(u)) = b = \mathrm{mot}_1(\omega_7)$.

9.2 Propriétés des automates à $\mathcal{P}_k$-piles

Les résultats prouvés dans cette section sont des résultats intermédiaires qui permettrons de prouver les théorème de la section suivante. Nous montrons que les automates à $\mathcal{P}_k$-piles sont simulables par différents types d'automates :

1. tout automate dans $\mathcal{P}_k$-$\mathsf{AP}^{\emptyset}$ est simulable par un automate dans k-$\mathsf{AP}^{\emptyset}$
2. tout automate dans $\mathcal{P}_k$-$\mathsf{AP}^{\emptyset}$ est simulable par un automate dans $\overline{k\text{-}\mathsf{AP}}^{\emptyset}$
3. tout automate dans $\mathcal{P}_k$-$\mathsf{AP}^{(\mathsf{REC}_{\ell+1})_{\ell+1}}$ est simulable par un automate dans k-$\mathsf{AP}^{(\mathsf{PREC}_{\ell})_{\ell}}$
4. tout automate dans $\mathcal{P}_{k+\ell}$-$\mathsf{AP}^{\emptyset}$ est simulable par un automate dans k-$\mathsf{AP}^{(\mathsf{REC}_{\ell+1})_1}$.

9.2.1 Expression des instructions

Nous montrons ici que les instructions $\mathrm{prod}_{i,a}$ peuvent être définies comme une composition d'instructions classiques, et comme une composition d'instructions dans $\overline{\mathfrak{I}nstr}_k$, dont la valeur ne dépend que des symboles de tête de la pile à laquelle elle est appliquée.

Regardons tout d'abord comment découper prod_a à l'aide d'instructions push et pop.

Lemme 9.2.1. *Pour tout $a \in A_i \cup \overline{A_i}$, $i \geq 1$, et tout $k \geq 1$, et $u \in \mathcal{P}_k$ tel que $u \bullet a \in \mathcal{P}_k$, et tout $\omega = \mathcal{C}(u)$:*

$$\begin{aligned}\mathrm{prod}_a(\omega) &= a[\mathrm{prod}_a(\mathrm{p}_{k-1}(\omega))] \cdot \omega \text{ si } i < k \text{ et } \mathrm{top}_k(\omega) \neq \overline{a} \\ &= \mathrm{push}_{k,a}(\omega) \text{ si } i = k \text{ et } \mathrm{top}_k(\omega) \neq \overline{a} \\ &= \mathrm{pop}_k(\omega) \text{ si } \mathrm{top}_k(\omega) = \overline{a}.\end{aligned}$$

Preuve : Distinguons deux cas selon la valeur de $\mathrm{top}_k(\omega)$:

- si $\mathrm{top}_k(\omega) \neq \overline{a}$, comme par définition, $\mathrm{mot}_k(\omega) = u$, alors $u \bullet a = u \cdot a$. Par définition de $\mathcal{C}_k$ et prod, nous avons donc

$$\mathrm{prod}_{k,a}(\omega) = a[(\mathcal{C}_{k-1}(\mathrm{f}_{k-1}(u \bullet a)))] \cdot \omega \tag{9}$$

  Distinguons à nouveau deux cas selon la valeur de i :
  - si $i = k$, alors $\mathrm{f}_{k-1}(u \bullet a) = \mathrm{f}_{k-1}(u)$. En substituant $\mathrm{f}_{k-1}(u \bullet a)$ par $\mathrm{f}_{k-1}(u)$ dans (9), et par définition de $\mathcal{C}_k$, nous obtenons donc

$$\mathrm{prod}_{k,a}(\omega) = \mathrm{push}_{k,a}(\omega),$$

  - si $1 \leq i < k$, alors $\mathrm{f}_{k-1}(u \bullet a) = \mathrm{f}_{k-1}(u) \bullet a \in \mathcal{P}_{k-1}$. Nous obtenons alors :

$$\begin{aligned}\mathcal{C}_{k-1}(\mathrm{f}_{k-1}(u \bullet a)) &= \mathcal{C}_{k-1}(\mathrm{f}_{k-1}(u) \bullet a) \\ &= \mathrm{prod}_{k-1,a}(\mathcal{C}_{k-1}(\mathrm{f}_{k-1}(u))) \\ &= \mathrm{prod}_{k-1,a}(\mathrm{p}_{k-1}(\omega)) \text{ par définition de } \mathcal{C}_k.\end{aligned}$$

    En substituant ce résultat dans (9), nous obtenons

$$\mathrm{prod}_a(\omega) = a[\mathrm{prod}_a(w_1)] \cdot \omega$$

- si $\mathrm{top}_k(\omega) = \overline{a}$, alors, il existe $u' \in \mathcal{P}_k$ tel que $u = u'\overline{a}$ (car $\mathrm{mot}_k(\omega) = u$). Alors $u \bullet a = u'$ et par définition de $\mathcal{C}_k$, $\omega = \overline{a}[\mathcal{C}_{k-1}(\mathrm{f}_{k-1}(u))] \cdot \mathcal{C}_k(u')$, d'où

$$\mathrm{prod}_a(\omega) = \mathrm{pop}_k(\omega).$$

□

Remarque 9.2.2. *Définissons pour tout $a \in A_i \cup \overline{A_i}$, $i \geq 1$, et $w = a_k \cdots a_1$, $k \geq 0$ l'instruction* $\mathrm{instr}_{w,a} \in \mathfrak{I}nstr_k^*$ *suivante :*

– *si $a_k \neq \overline{a}, \ldots, a_i \neq \overline{a}$, alors*

$$\mathrm{instr}_{w,a} = \mathrm{push}_{k,a} \cdots \mathrm{push}_{i,a}$$

– *si $a_j = \overline{a}$, $1 \leq j \leq i$ et $a_k \neq \overline{a}, \ldots, a_{j+1} \neq \overline{a}$, alors*

$$\mathrm{instr}_{w,a} = \mathrm{push}_{k,a} \cdots \mathrm{push}_{j+1,a}\, \mathrm{pop}_j$$

D'après le lemme précédent, il est clair que pour tout $\omega \in \mathcal{P}_k$-Pile tel que prod_a est défini et $\mathrm{top}(\omega) = w$,

$$\mathrm{instr}_{w,a}(\omega) = \mathrm{prod}_a(\omega)$$

Lemme 9.2.3. *Pour tout $a \in A_i \cup \overline{A_i}$, $i \geq 1$, et $u \in \mathcal{P}_k$, $k \geq 1$, tel que $u \bullet a \in \mathcal{P}_k$, et tout $\omega = \mathcal{C}(u) = a_k[\omega_1]\omega'$:*

$$\begin{aligned} \mathrm{prod}_a(\omega) &= a[\mathrm{prod}_a(\omega_1)] \cdot \omega \text{ si } i < k \text{ et } a_k \neq \overline{a} \\ &= \mathrm{push}_{k,a}(\omega) \text{ si } i = k \text{ et } a_k \neq \overline{a} \\ &= \overline{\mathrm{push}}_{k,\overline{a}}(a_k[\mathrm{prod}_a(\omega_1)] \cdot \omega') \text{ si } i < k \text{ et } a_k = \overline{a} \end{aligned}$$

Preuve : Les cas $a_k \neq \overline{a}$ sont identiques à ceux du lemme 9.2.1, et donc vrais. Considérons donc $\omega = \overline{a}[\omega_1]\omega'$ avec $\omega = \mathcal{C}(u)$ et $u \bullet a \in \mathcal{P}_k$, $k \geq i$. En procédant comme dans le second point de la preuve du lemme 9.2.1 nous obtenons encore $u = u'a$, $\mathcal{C}(u') = \omega'$, et $\mathrm{prod}_{k,a}(\omega) = \omega'$ Alors,

$$\overline{a}[\mathrm{prod}_a(\omega_1)]\omega' = \overline{a}[\mathrm{prod}_a(\mathcal{C}_{k-1}(\mathrm{f}_{k-1}(u)))]\omega' = \overline{a}[\mathcal{C}_{k-1}(\mathrm{f}_{k-1}(u \bullet a))]\omega'$$

– si $k = i$,
alors, $\mathcal{C}_{k-1}(\mathrm{f}_{k-1}(u \bullet a) = \mathcal{C}_{k-1}(\mathrm{f}_{k-1}(u)) = \mathrm{p}_{k-1}(\omega')$ et donc, $\overline{\mathrm{push}}_{k,\overline{a}}(\overline{a}[\mathrm{prod}_a(\omega_1)]\omega')$ est défini et égal à $\mathrm{prod}_a(\omega)$,
– sinon

$$\mathcal{C}_{k-1}(\mathrm{f}_{k-1}(u \bullet a) = \mathcal{C}_{k-1}(\mathrm{f}_{k-1}(u')) = \mathrm{p}_{k-1}(\omega')$$

Donc $\overline{\mathrm{push}}_{k,\overline{a}}(\overline{a}[\mathrm{prod}_a(\omega_1)]\omega')$ est défini et égal à $\mathrm{prod}_a(\omega)$.
□

Remarque 9.2.4. *Définissons pour tout $a \in A_i \cup \overline{A_i}$, $i \geq 1$, et $w = a_k \cdots a_1$, $k \geq 0$ l'instruction* $\overline{\mathrm{instr}}_{a,w} \in \mathfrak{I}nstr_k^*$ *suivante : Soit $k_1 > \cdots > k_m$ la suite des indices $j \in [i,k]$ tels que $a_j \neq \overline{a}$ et $l_1 < \cdots < l_n$ la suite complémentaire dans $[i,k]$, alors*

$$\overline{\mathrm{instr}}_{w,a} = \mathrm{push}_{k_1,a} \cdots \mathrm{push}_{k_m,a}\, \overline{\mathrm{push}_{l_1,a}} \cdots \overline{\mathrm{push}_{l_1,a}}.$$

D'après le lemme précédent, il est clair que pour tout $\omega \in \mathcal{P}_k$-Pile tel que prod_a est défini et $\mathrm{top}(\omega) = w$,

$$\overline{\mathrm{instr}}_{w,a}(\omega) = \mathrm{prod}_a(\omega).$$

9.2.2 Simulation par un automate à k-piles

Proposition 9.2.5. *Pour tous alphabets de piles $A_1, \ldots, A_k$, pour tout vecteur $\vec{C}$, de sous-ensembles de k-*$\mathsf{Pile}(A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}})$*,*

$$\mathcal{P}_k\text{-}\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k) \quad \sqsubseteq_{id} \quad k\text{-}\mathsf{AP}^{\vec{C}}(A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}})$$

Preuve : Soit $\mathcal{M} = (Q, \Sigma, (A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}}), \vec{C}, \Delta, q_0, F) \in \mathcal{P}_k\text{-}\mathsf{AP}(A_1, \ldots, A_k)$. Nous construisons l'automate $\mathcal{A} = (Q, \Sigma, (A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}}), \vec{C}, \Delta', q_0, F) \in {}^*k\text{-}\mathsf{AP}(A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}})$ tel que $\mathrm{id} : \omega \mapsto \omega$ est une simulation déterministe de $\mathcal{M}$ par $\mathcal{A}$. En utilisant la remarque 9.2.2, il suffit de définir Δ' comme l'ensemble des transitions

$$\Delta'_\delta = \{(p, \alpha, a_k \cdots a_1, \vec{o}, \mathrm{instr}_{a_k \cdots a_1, a}, q)\}$$

pour $\delta = (p, \alpha, a_k \cdots a_1, \vec{o}, \mathrm{prod}_{k,a}, q) \in \Delta$

L'application id est clairement une simulation déterministe de $\mathcal{M}$ par $\mathcal{A}$.

Nous avons donc $\mathcal{P}_k\text{-}\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k) \sqsubseteq_{id} {}^*k\text{-}\mathsf{AP}^{\vec{C}}(A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}})$ et nous avons déjà remarqué que ${}^*k\text{-}\mathsf{AP}^{\vec{C}}(A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}}) \sqsubseteq_{id} k\text{-}\mathsf{AP}^{\vec{C}}(A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}})$.

En appliquant la loi de composition (lemme 7.1.8), la proposition est démontrée.

□

Remarque 9.2.6. *L'automate construit est N2-normalisé puisque aucune instruction* change *n'est utilisée, nous avons donc $\mathcal{P}_k\text{-}\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k) \sqsubseteq_{\mathrm{id}} k\text{-}\mathsf{N2AP}^{\vec{C}}(A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}})$*

Lemme 9.2.7. *Pour tout $R \in \mathsf{REC}_k(A_1, \ldots, A_k)$, $k \geq 0$,*

$$\mathcal{C}_k(R) \in \mathsf{PREC}_k(A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}}).$$

Preuve : Prouvons ce lemme par induction sur $k \geq 0$:
Base : si $k = 0$, alors $\mathsf{REC}_0 = \{\{\varepsilon\}, \emptyset\}$ $\mathsf{PREC}_0 = \{\{\bot_0\}, \emptyset\}$, comme $\mathcal{C}_0(\varepsilon) = \bot_0$, la propriété est vérifiée.
Pas d'induction : supposons le lemme vrai pour $k \geq 0$. Soit $R \in \mathsf{REC}_{k+1}$, il existe un vecteur $\vec{O}$ d'éléments de REC_k, et un automate fini $\mathcal{A}$ à oracle ${\mathrm{f}_k}^{-1}(\vec{O})$ déterministe et complet dans $\mathcal{P}_{k+1}$, reconnaissant le langage R. Par une construction similaire à celle donnée dans la preuve du lemme 9.1.8, nous obtenons une machine dans $\mathcal{P}_{k+1}\text{-}\mathsf{AP}^{(\vec{O})_k}$ générant le langage de pile $\mathcal{C}_{k+1}(R)$. En utilisant la proposition 9.2.5 et les propriétés de la simulation déterministe, $\mathcal{C}_{k+1}(R)$ est donc le langage de piles généré par un automate dans $(k+1)\text{-}\mathsf{AP}^{[\mathcal{C}_k(\vec{O})]_k}$.

Par hypothèse d'induction $\mathcal{C}_k(\vec{O}) \in \mathsf{PREC}_k$, et en utilisant le théorème 6.2.7, $\mathcal{C}_{k+1}(R)$ appartient donc à PREC_{k+1}.

□

9.2.3 Simulation par un automate à instructions symétriques

Lemme 9.2.8.

$$\mathcal{P}_k\text{-}\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k) \sqsubseteq_{id} \overline{k\text{-}\mathsf{AP}}^{\vec{C}}(A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}}).$$

Preuve : Soit $\mathcal{M} = (Q, \Sigma, (A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}}), \vec{C}, \Delta, q_0, F) \in \mathcal{P}_k\text{-}\mathsf{AP}(A_1, \ldots, A_k)$. Nous construisons l'automate $\mathcal{A} = (Q, \Sigma, (A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}}), \vec{C}, \Delta', q_0, F) \in {}^*\overline{k\text{-}\mathsf{AP}}(A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}})$ tel que $id : \omega \mapsto \omega$ est une simulation déterministe de $\mathcal{M}$ par $\mathcal{A}$. En utilisant la remarque 9.2.4, il suffit de définir Δ' comme l'union des systèmes

$$\Delta'_\delta = \{(p, \alpha, a_k \cdots a_1, \vec{o}, \overline{\mathrm{instr}}_{a_k \cdots a_1, a}, q)\}$$

pour $\delta = (p, \alpha, a_k \cdots a_1, \vec{o}, \mathrm{prod}_{k,a}, q) \in \Delta$.

□

9.2.4 Simulation par un automate à contrôleur de niveau inférieur

Étudions maintenant le cas où le mot de niveau $\ell + 1$ d'un $\mathcal{P}_k$-AP est contrôlé par un langage $(\ell+1)$-régulier. Par définition, tout langage $R \in \mathsf{REC}_{\ell+1}$ est reconnu par un automate fini $\mathcal{A}$ à oracles ${\mathrm{f}_\ell}^{-1}(\vec{O})$, où $\vec{O}$ est un vecteur de langages ℓ-réguliers. Alors, tout automate dans $\mathcal{P}_k\text{-}\mathsf{AP}^{(R)_{\ell+1}}$ peut être simulé par un automate dans ${}^*k\text{-}\mathsf{AP}^{(\vec{O})_\ell}$. Il suffit pour cela de reprendre le même type de construction que pour la proposition 9.2.5, mais en faisant en plus le produit du niveau $\ell + 1$ de la pile avec l'automate $\mathcal{A}$, en substituant les tests des oracles effectués de $\mathcal{A}$ par des contrôles sur le niveau ℓ de la pile.

Précisons tout d'abord la simulation que nous allons utiliser : soit B un alphabet fini et $\eta : A_\ell^* \to B$ une fonction quelconque définie pour ε. Nous définissons pour tout $k \geq 0$, l'application

$$\mathrm{id}_\eta : k\text{-}\mathsf{Pile}(A_1, \ldots, A_k) \to k\text{-}\mathsf{Pile}^{(\bot, \eta(\varepsilon))}(A_1, \ldots, A_{\ell-1}, A_\ell \times B, A_{\ell+1}, \ldots A_k)$$

transformant chaque $\omega \in k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$ en une k-pile quasi-identique, hormis les étiquettes de niveau ℓ :

si $\mathrm{mot}_\ell(\omega) = a_{\ell,1} \cdots a_{\ell,n}$, alors $\mathrm{mot}_\ell(\mathrm{id}_\eta(\omega)) = (a_{\ell,1}, \eta(a_{\ell,1})) \cdots (a_{\ell,n}, \eta(a_{\ell,1} \cdots a_{\ell,n}))$. La figure 21 illustre cette transformation.

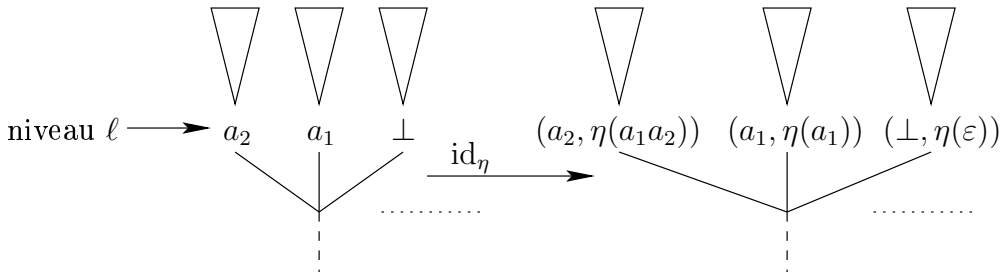


FIG. 21 – Application de la fonction id_η

Posons par convention $\mathrm{id}_\eta(\varepsilon) = \varepsilon$; pour tout $\omega = a[\omega_1]\omega' \in k\text{-}\mathsf{Pile}$,
- $\mathrm{id}_\eta(\omega) = (a, \eta(u))[\mathrm{id}_\eta(\omega_1)]\,\mathrm{id}_\eta(\omega')$, si $k = \ell$ et $u = \mathrm{mot}_\ell(\omega)$ et $\eta(u)$ est défini
- $\mathrm{id}_\eta(\omega) = a[\mathrm{id}_\eta(\omega_1)]\,\mathrm{id}_\eta(\omega')$, si $k \neq \ell$
- $\mathrm{id}_\eta(\omega)$ est indéfini si $k = \ell$ et $\eta(\mathrm{mot}_k(\omega))$ est indéfini.

Remarque 9.2.9. *Une induction évidente sur $k \geq 1$ permet de vérifier que pour tout $\omega \in k$-Pile, pour tout $i \geq 1$*

- $\mathrm{id}_\eta(\mathrm{pop}_i(\omega)) = \mathrm{pop}_i(\mathrm{id}_\eta(\omega))$
- *si $i \neq k$,* $\mathrm{id}_\eta(\mathrm{push}_{i,a}(\omega)) = \mathrm{push}_{i,a}(\mathrm{id}_\eta(\omega))$
- *si $i = k$, alors* $\mathrm{id}_\eta(\mathrm{push}_{i,a}(\omega)) = \mathrm{push}_{\ell,(a,\eta(\mathrm{mot}_\ell(\omega)))}(\mathrm{id}_\eta(\omega))$.

Proposition 9.2.10. *Pour tous $0 \leq \ell < k$,*

$$\mathcal{P}_k\text{-}\mathsf{AP}^{(\mathsf{REC}_{\ell+1})_{\ell+1}} \sqsubseteq k\text{-}\mathsf{AP}^{[\mathsf{PREC}_\ell]_\ell}$$

En d'autres mots, pour tous alphabets de piles $A_1, \ldots, A_k$, et tout vecteur $\vec{R}$ d'ensembles dans $\mathsf{REC}_{\ell+1}(A_1, \ldots, A_{\ell+1})$, il existe des alphabets de piles $B_1, \ldots, B_k$ et un vecteur $\vec{D}$ d'ensembles dans $\mathsf{PREC}_\ell(B_1, \ldots, B_\ell)$ tels que

$$\mathcal{P}_k\text{-}\mathsf{AP}^{(\vec{R})_{\ell+1}}(A_1, \ldots, A_k) \quad \sqsubseteq \quad k\text{-}\mathsf{AP}^{[\vec{D}]_\ell}(B_1, \ldots, B_k).$$

Construction : Supposons tout d'abord pour simplifier, que $\vec{R} = R \in \mathsf{REC}_{\ell+1}(A_1, \ldots, A_{\ell+1})$. Soit

$$\mathcal{M} = (Q, \Sigma, (A_1^{\mathcal{C}}, \ldots, A_k^{\mathcal{C}}), (R)_{\ell+1}, \Delta, q_0, F) \in \mathcal{P}_k\text{-}\mathsf{AP}(A_1, \ldots, A_k)$$

Par définition de $\mathsf{REC}_{\ell+1}$, il existe un vecteur $\vec{O}$ de $m \geq 0$ ensembles dans $\mathsf{REC}_\ell(A_1, \ldots, A_\ell)$, et un automate fini $\mathcal{A} \in \mathsf{AF}_{\ell+1}(A_1, \ldots, A_{\ell+1})$, à oracles $\mathrm{f}_\ell^{-1}(\vec{O})$, reconnaissant le langage R. Supposons $\mathcal{A} = (Q_\mathcal{A}, \Delta_\mathcal{A}, \mathrm{f}_\ell^{-1}(\vec{O}), p_0, F_\mathcal{A})$ déterministe et complet dans $\mathcal{P}_{\ell+1}$.

Posons $B_i = A_i^{\mathcal{C}}$ pour tout $j \neq \ell + 1$ et $B_{\ell+1} = A_{\ell+1}^{\mathcal{C}} \times Q_\mathcal{A}$, et

$$\mathcal{A} = (Q, \Sigma, (B_1, \ldots, B_k), [\mathcal{C}_\ell(\vec{O})]_\ell, \Delta', q_0, F, (\bot, p_0)) \in {}^*k\text{-}\mathsf{AP}(B_1, \ldots, B_k)$$

où Δ' est l'union des systèmes de transitions Δ'_δ, pour $\delta \in \Delta$ et tels que

$$\text{si } \delta = (q, \alpha, a_k \cdots a_1, b, \mathrm{prod}_{k,a}, q') \text{ avec } a \in A_i \cup \overline{A}_i$$

alors

- si $\mathrm{instr}_{a_k \cdots a_1, a}$ contient ne contient pas l'instruction $\mathrm{push}_{\ell+1,a}$, alors Δ'_δ est l'ensemble des transitions
$$(q, \alpha, a_k \cdots (a_{\ell+1}, p) \cdot a_\ell \cdots a_1, \vec{o'}, \mathrm{instr}_{a_k \cdots a_1, a}, q')$$
pour $\vec{o'} \in \{0,1\}^m$ et $\chi_{F_\mathcal{A}}(p) = b$
- sinon Δ'_δ est l'ensemble des transitions
$$(p, \alpha, a_k \cdots (a_{\ell+1}, p) \cdot a_\ell \cdots a_1, \vec{o}, \mathrm{instr}_{a_k \cdots a_1, a, p'}, q')$$
pour $(p, a, \vec{o}, p') \in \Delta_\mathcal{A}$ et $\chi_{F_\mathcal{A}}(p) = b$. ($\mathrm{instr}_{w,a,p'}$ désigne l'instruction $\mathrm{instr}_{w,a}$ dans laquelle l'instruction $\mathrm{push}_{\ell+1,a}$ est remplacée par $\mathrm{push}_{\ell+1,(a,p')}$).

D'après le lemme 9.2.7, puisque les composantes de $\vec{O}$ appartiennent à REC_ℓ, alors les composantes de $\mathcal{C}_\ell(\vec{O})$ appartiennent à PREC_ℓ.

Preuve : Par choix de $\mathcal{A}$ complet déterministe, il existe une application $\eta : \mathcal{P}_{\ell+1} \to Q_\mathcal{A}$ associant à tout u l'unique p tel que $(p_{0,\uparrow}\, u) \vdash_\mathcal{A}^* (p, u_\uparrow)$. Alors, $u \in R$ ssi $\eta(u) \in F$. Vérifions que $\mathcal{M} \leq_{\mathrm{id}_\eta} \mathcal{A}$.

Nous montrons que pour toute transition $\delta \in \Delta$, pour tout $u \in \mathcal{P}_k$ ${\delta^{(R)}}_{\ell+1} \leq_{\mathrm{id}_\eta, \mathcal{C}_k(u)} {\Delta'_\delta}^{(\vec{O})_\ell}$. Supposons

$$\delta = (q, \alpha, a_k \cdots a_1, b, \mathrm{prod}_{k,a}, q') \text{ avec } a \in A_i \cup \overline{A}_i.$$

Alors, pour tous $\omega, \omega' \in \mathcal{P}_k$-Pile, avec $\mathrm{mot}_{\ell+1}(\omega) = u$

$$\begin{aligned} & (p, \sigma, \omega) \vdash_{{\delta^{(R)}}_{\ell+1}} (q, \varepsilon, \omega') \qquad (10) \\ \text{ssi} \quad & \mathrm{top}(\omega) = a_k \cdots a_1 \text{ et } \chi_R(u) = b \text{ et } \omega' = \mathrm{prod}_a(\omega) \\ \text{ssi} \quad & \mathrm{top}(\mathrm{id}_\eta(\omega)) = a_k \cdots (a_{\ell+1}, \eta(u)) \cdots a_1,\ \chi_{F_\mathcal{A}}(\eta(u)) = b \text{ et } \mathrm{id}_\eta(\omega') = \mathrm{id}_\eta(\mathrm{prod}_a(\omega)). \end{aligned}$$

Étudions deux cas selon la valeur de $\mathrm{instr}_{a_k \cdots a_1, a}$:

- si $\mathrm{instr}_{a_k \cdots a_1, a}$ ne contient pas l'instruction $\mathrm{push}_{\ell+1,a}$, alors d'après la remarque 9.2.9, nous avons donc $\mathrm{instr}_{a_k \cdots a_1, a}(\mathrm{id}_\eta(\omega)) = \mathrm{id}_\eta(\omega)$, d'où

$$(10) \quad \text{ssi} \quad (q, \mathrm{id}_\eta(\omega)) \vdash_{{\Delta'_\delta}^{[\vec{C}]_\ell}} (q', \mathrm{id}_\eta(\omega'))$$

- sinon $u \bullet a = u \cdot a \in \mathcal{P}_{\ell+1}$ et la transition $(p, a, \chi_{\vec{O}}, \eta(ua))$ appartient à $\Delta_\mathcal{A}$ et en utilisant la remarque 9.2.9, $\mathrm{instr}_{a_k \cdots a_1, \eta(ua)}(\mathrm{id}_\eta(\omega)) = \mathrm{id}_\eta(\omega')$. D'où

$$(10) \quad \text{ssi} \quad (q, \mathrm{id}_\eta(\omega)) \vdash_{{\Delta'_\delta}^{[\vec{C}]_\ell}} (q', \mathrm{id}_\eta(\omega')).$$

□

9.2.5 Simulation par un automate de niveau inférieur

Une caractéristique très forte des $\mathcal{P}_k$-piles est que, pour tout $1 \leq \ell < k$, le niveau ℓ de la pile est complètement déterminé par ses niveaux supérieurs, et même simplement par les symboles apparaissant au niveau $\ell + 1$.

En effet, pour tout $\omega \in \mathcal{P}_k$-Pile, si $\mathrm{mot}_{\ell+1}(\omega) = u$, alors $u \in \mathcal{P}_{\ell+1}$ et $\mathrm{p}_\ell(\omega) = \mathcal{C}_\ell(f_\ell(u))$ (voir remarque 9.1.3). Cette propriété est illustrée par la figure 22. Ainsi, au cours d'un cal-

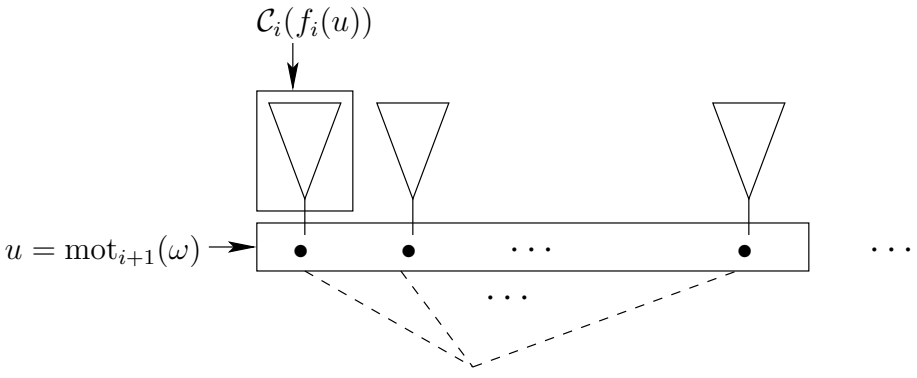


FIG. 22 – $\mathrm{p}_i(\omega) = \mathcal{C}_i(f_i(\mathrm{mot}_{i+1}(\omega)))$

cul dans un $\mathcal{P}_k$-AP, le niveau ℓ de la pile courante n'est réellement utile que pour effectuer les tests $\mathrm{top}_{\ell \cdots 1}(\omega)$. Il est donc possible de simuler l'exécution d'un $\mathcal{P}_{k+\ell}$-AP par un k-AP contrôlé à l'aide une fonction envoyant chaque ω sur $\omega^{-\ell}$, obtenu en tronquant dans ω les

niveaux inférieurs à ℓ. L'information manquante $\mathrm{top}_{\ell\cdots 1}(\omega)$ sera reconstituée par un contrôle sur $\mathrm{mot}_{\ell+1}(\omega)$.

Formalisons tout d'abord la notion de pile "tronquée".

Définition 9.2.11. *Pour tous $\ell, k \geq 0$, l'application*

$$\mathrm{id}_{k+\ell}^{-\ell} : (k+\ell)\text{-}\mathsf{Pile}(A_1, \ldots, A_{k+\ell}) \to k\text{-}\mathsf{Pile}(A_{1+\ell}, \ldots, A_{k+\ell})$$

est définie pour tout $\omega \in (k+\ell)$-Pile par :

- *si $k = 0$, alors $\mathrm{id}_k^{-\ell}(\omega) = \perp_0$,*
- *si $k > 0$, alors ω s'écrit $\omega = a[\omega_1]\omega'$. Posons par convention $\mathrm{id}_k^{-\ell}(\varepsilon) = \varepsilon$.*
  Alors $\mathrm{id}_{k+\ell}^{-\ell}(\omega) = a[\mathrm{id}_{k+\ell-1}^{\ell-1}(\omega_1)]\,\mathrm{id}_{k+\ell}^{-\ell}(\omega')$.

Pour $\omega \in k+\ell$-Pile, nous noterons simplement $\omega^{-\ell}$ pour $\mathrm{id}_{k+\ell}^{-\ell}(\omega)$. Clairement, la restriction de $\mathrm{id}_{k+\ell}^{-\ell}$ aux $\mathcal{P}_k$-piles est injective.

Pour toute suite d'instructions dans $\mathfrak{I}nstr_k$, notons $(\mathrm{instr}_1 \cdots \mathrm{instr}_n)^{-\ell} = \mathrm{instr}_1^{-\ell} \cdots \mathrm{instr}_n^{-\ell}$ et

- $\mathrm{push}_{i,a}^{-\ell} = \mathrm{push}_{i-l,a}$ si $i > l$ et $\mathrm{push}_{i,a}^{-\ell} = \mathrm{id}$ sinon,
- $\mathrm{pop}_{i,a}^{-\ell} = \mathrm{pop}_{i-l,a}$ si $i > l$ et $\mathrm{pop}_{i,a}^{-\ell} = \mathrm{id}$ sinon.

Proposition 9.2.12.

$$\mathcal{P}_{k+\ell}\text{-}\mathsf{AP}^{\emptyset} \sqsubseteq k\text{-}\mathsf{AP}^{(\mathsf{REC}_{\ell+1})_1}$$

Preuve : Les tests $\mathrm{top}_i(\omega)$ pour $i \leq \ell$ seront remplacés par un contrôle sur le niveau 1 de ω^{-l}.

Le vecteur de contrôleur $\vec{T}$ que nous choisissons d'utiliser, est défini en fonction de ℓ et des alphabets de piles. Pour $\ell \geq 0$, et les alphabets de piles $A_1, \ldots, A_{\ell+1}$, les composantes de $\vec{T}$ sont tous les $T_{b_\ell \cdots b_1}$, pour $b_i \in A_i^{\mathcal{C}}$ vérifiant

$$T_{b_\ell \cdots b_1} = \{u \in \mathcal{P}_{\ell+1} \mid \mathrm{top}_{\ell\cdots 1}(\mathcal{C}_\ell(\mathrm{f}_\ell(u)) = b_\ell \cdots b_1\}.$$

Chaque $T_{b_\ell \cdots b_1}$ appartient à $\mathsf{REC}_{\ell+1}$, puisqu'il correspond (d'après la remarque 9.1.3(1)) à l'intersection pour $i \in [1, \ell]$ des ensembles ${\mathrm{f}_{\ell+1,i}}^{-1}(\mathcal{P}_i \cdot b_i)$ si $b_i \neq \perp$ et ${\mathrm{f}_{\ell+1,i}}^{-1}(\mathcal{P}_i)$ sinon.

Supposons le vecteur $\vec{T}$ orienté par une bijection $\beta : (A_\ell^{\mathcal{C}} \cup \perp) \cdots (A_1^{\mathcal{C}} \cup \perp) \to [1, m]$ quelconque. Nous associons chaque transition $\delta = (p, \alpha, a_{k+\ell} \cdots a_1, \mathrm{prod}_{k+\ell,a}, q)$ d'un $\mathcal{P}_{k+l}$-AP au système de transitions

$$\Delta'_\delta = \{(p, \alpha, a_{k+\ell} \cdots a_{k+1}, \vec{o}, \mathrm{instr}_{a_k \cdots a_1, a}^{-l}, q) \mid \pi_{\beta_\ell(a_i,i)}(\vec{o}) = 1,\ \forall i \in [1, \ell]\}.$$

Pour tout $\omega = \mathcal{C}_k(u)$, $\mathrm{top}_i(\omega) = a_i$ ssi $\pi_{\beta_\ell(a_i,i)}(\vec{o}) = 1$.
De plus $\mathrm{instr}_{a_k \cdots a_1, a}^{-l}(\omega^{-l}) = (\mathrm{instr}_{a_k \cdots a_1, a}(\omega))^{-l}$, nous avons donc clairement

$$\delta \leq_{\mathrm{id}_{k+l}^{-l}, \omega} {\Delta'_\delta}^{(\vec{T})_1}.$$

□

9.3 Transfert sur les automates à k-piles

9.3.1 Simulation d'un automate à k-piles

Nous prouvons dans cette section que tout automate à k-piles est simulable par un automate à $\mathcal{P}_k$-piles. Puisque nous disposons déjà de la bijection φ_k, de k-Pile dans $\mathcal{P}_k$, la simulation que nous choisissons d'utiliser est l'application bijective

$$\mathcal{D}_k = \varphi_k \circ \mathcal{C}_k : k\text{-Pile}(A_1, \ldots, A_k) \to \mathcal{P}_k\text{-Pile}(A_1, \ldots, A_k)$$

En appliquant la propriété de composition de la simulation déterministe, et les équivalences énoncées dans le théorème 8.2.4, nous pouvons nous contenter de montrer k-NAP $\sqsubseteq \mathcal{P}_k$-AP.

Avant de voir comment simuler les instructions classiques, intéressons nous aux symboles de têtes. Étant donné $\omega \in k$-Pile, nous pouvons facilement remarquer que le mot $\mathrm{top}(\omega)$ n'est pas directement identifiable à partir de $\mathrm{top}(\mathcal{D}_k(\omega))$. Considérons par exemple, $\omega = \bot\ [\bot\ [a_1\ \bot]]$, dans ce cas $\mathcal{D}_3(\omega) = a_1[a_1[a_1\ \bot]\ \bot]\ \bot_3$ et, hormis la donnée de $\mathrm{top}_1(\omega)$, le mot $\mathrm{top}(\mathcal{D}_3(\omega))$ ne semble pas contenir d'informations pertinentes sur la valeur de $\mathrm{top}(\omega)$.

Toutefois, il existe des cas particuliers où $\mathrm{top}(\omega)$ peut être obtenu directement des symboles de têtes de $\mathcal{D}(\omega)$.

Lemme 9.3.1. *Soit $\omega \in k$-NPile$(A_1, \ldots, A_k)$, avec $A_2 = \{\#_2\}, \ldots, A_k = \{\#_k\}$, dans ce cas,*
- *pour tout $i \geq 2$, $\mathrm{top}_i(\omega) = \#_i$ ssi $\mathrm{top}_i(\mathcal{D}_k(\omega)) \neq \bot$,*
- $\mathrm{top}_1(\omega) = \mathrm{top}_1(\mathcal{D}_k(\omega))$.

Preuve : Nous montrons tout d'abord que pour $A_2, \ldots, A_k$ quelconques, pour tout $\omega \in k$-NPile$(A_1, \ldots, A_k)$ et $i \geq 2$, alors

$$\mathrm{top}_i(\omega) = \bot \text{ ssi } \mathrm{top}_i(\mathcal{D}_k(\omega)) = \bot\ . \tag{11}$$

Vérifions cette propriété par induction sur $k \geq 1$.
Base : traitons le cas $k = 1$, dans le cas plus général où $i = k$.

Par définition de piles N-normalisées, puis de $\mathcal{C}_k$, nous avons :

$$\mathrm{top}_k(\omega) = \bot \text{ ssi } \omega = \bot_k \text{ ssi } \varphi_k(\omega) = \varepsilon \text{ ssi } \mathrm{top}_k(\mathcal{D}_k(\omega)) = \bot\ \mathcal{C}_k(\varphi_k(\omega)) = \bot_k\ .$$

En particulier, si $k = 1$, alors $i = 1$ et la propriété est vérifiée.
Pas d'induction : supposons la propriété vraie pour $k \geq 1$.

Soit $\omega = a[\omega_1]\omega' \in (k+1)$-NPile$(A_1, \ldots, A_{k+1})$, et $i \in [1, k]$. Alors,
$\mathrm{top}_i(\omega) = \bot$ ssi $\mathrm{top}_i(\omega_1) = \bot$ ssi (par h.r.) $\mathrm{top}_i(\mathcal{D}_k(\omega_1)) = \bot$ ssi (par définition de $\mathcal{C}_{k+1}$) $\mathrm{top}_i(\mathcal{D}_{k+1}(\omega)) = \bot$.
(fin de l'induction)

Si de plus, les alphabets de piles $A_2, \ldots, A_k, \ldots$ sont réduits à un unique élément, alors la propriété que nous venons de prouver implique clairement que pour tout $1 \leq i \leq k$, $\mathrm{top}_i(\omega) = \#_i$ ssi $\mathrm{top}_i(\mathcal{C}(\varphi_k(\omega))) \neq \bot$. De plus,

$$\begin{aligned} \mathrm{mot}_1(\omega) &= \mathrm{f}_{k,1}(\varphi_k(\omega)) \\ &= \mathrm{mot}_1(\mathcal{C}_k(\varphi_k(\omega))) \text{ (par définition de } \mathcal{C}_k \text{)} \\ &= \mathrm{mot}_1(\mathcal{D}_k(\omega)) \end{aligned}$$

En particulier, $\mathrm{top}_1(\omega) = \mathrm{top}_1(\mathcal{D}_k(\omega))$.

□

Prouvons maintenant que pour toute instruction δ d'un automate dans k-NAP avec des alphabets de niveau supérieur à 1 réduits à un élément, et tout vecteur $\vec{R}$ de contrôleurs, $\delta^{\vec{C}}$ est simulable par un système dans $\mathcal{P}_k$-AP.

Il faut pour cela être capable pour chaque $\omega \in k\text{-Pile}(A_1, \ldots, A_k)$, pour tout instr de type push ou pop, de définir une suite d'instructions de type prod, envoyant $\mathcal{D}_k(\omega)$ sur $\mathcal{D}_k(\text{instr}(\omega))$, Nous utilisons pour cela les résultats obtenus au §6.1.1 sur les liens entre l'application des instructions de piles et le produit à droite dans $\mathcal{P}_k$.

Si l'instruction de pile est du type $\text{push}_{i,a}$, d'après la proposition 6.1.1, $\varphi_k(\text{push}_{i,a}(\omega))$ est obtenu de $\varphi(\omega)$ par produit avec la lettre a :

Lemme 9.3.2. *Soit $a \in A_i$, $1 \leq i \leq k$ et $\omega \in k$-Pile$(A_1, \ldots, A_k)$, alors*
$\varphi_k(\text{push}_{i,a}(\omega)) = \varphi_k(\omega) \bullet a$

Donc, par définition de l'instruction prod_a : $\mathcal{D}_k(\text{push}_{i,a}(\omega)) = \text{prod}_a(\mathcal{D}_k(\omega))$

Proposition 9.3.3. *Pour toute transition $\delta = (p, \alpha, w, \vec{o}, \text{push}_{i,a}, q) \in k\text{-NAP}^{\vec{C}}(A_1, \ldots, A_k)$, où $A_2, \ldots, A_k$ sont réduits à un unique élément, il existe un système $\Delta_\delta \in \mathcal{P}_k\text{-AP}(A_1, \ldots, A_k)$, tel que pour tout $\omega \in k$-Pile $\delta^{\vec{C}} \leq_{\mathcal{D}_k,\omega} {\Delta_\delta}^{\mathcal{D}_k(\vec{C})}$.*

Preuve : Supposons que $w = b_k \cdots b_1$, nous définissons le système de transitions (p, α, q)-standard d'un $\mathcal{P}_k$-AP suivant :

– si $b_k \cdots b_1 \neq \perp^k$,

$$\Delta_\delta = \{(p, \alpha, a_k \cdots a_2 b_1, \vec{o}, \text{prod}_a, q) \mid \forall i \in [1, k],\ a_i \in A_i^{\mathcal{C}}\}$$

– si $b_j \cdots b_1 = \perp^j$ et $b_{j+1} \neq \perp$ alors

$$\Delta_\delta = \{(p, \alpha, a_k \cdots a_{j+1} \perp \cdots \perp, \vec{o}, \text{prod}_a, q) \mid \forall i \in [1, k],\ a_i \in A_i^{\mathcal{C}}\}$$

Ce système est clairement déterministe et d'après les trois lemmes précédents, il est évident que pour tout $\omega \in k$-NPile$(A_1, \ldots, A_k)$,

$$(p, \alpha, \omega) \vdash_{\delta^{\vec{C}}} (q, \varepsilon, \text{push}_a(\omega)) \text{ ssi } (p, \alpha, \mathcal{D}_k(\omega)) \vdash^*_{{\Delta_\delta}^{\mathcal{D}_k(\vec{C})}} (q, \varepsilon, \mathcal{D}_k(\text{push}_a(\omega))),$$

et $\delta^{\vec{C}}$ est applicable à ω ssi ${\Delta_\delta}^{\mathcal{D}_k(\vec{C})}$ est applicable à $\mathcal{D}_k(\omega)$.

□

Le calcul de $\mathcal{D}_k(\text{pop}_i(\omega))$ est légèrement plus compliqué puisque d'après le lemme 6.1.3, $\varphi_k(\text{pop}_i(\omega))$ est obtenu par produit de $\varphi_k(\omega)$ avec un mot $u \in {A_{1,i}}^*$.

Toutefois, u ne dépend que de la valeur de $\varphi_k(\text{f}_{k,i}(\omega))$, c'est à dire de la valeur de $\text{mot}_{k,i}(\mathcal{D}_k(\omega))$. Rappelons ici le lemme 6.1.3 :

Lemme 9.3.4. *Soit $\omega, \omega' \in k$-Pile$(A_1, \ldots, A_k)$, et $i \in [1, k]$ tel que $\omega' = \text{pop}_i(\omega)$,*
alors il existe $a \in A_i$ tel que $\text{top}_i(\omega) = a$ et $u \in \text{Irr}(A_{1,i-1})$ tels que
$\text{f}_{k,i}(\varphi_k(\omega)) \in \mathcal{P}_i \cdot a \cdot u$ *et* $\varphi_k(\omega') = \varphi_k(\omega) \bullet \overline{u} \bullet \overline{a}$.

Lemme 9.3.5. *Pour toute transition $\delta(p, \alpha, b_k \cdots b_1, \vec{o}, \text{pop}_i, q) \in k\text{-NAP}^{\vec{C}}(A_1, \ldots, A_k)$, où les alphabets $A_2, \ldots, A_k$ sont réduits à un unique élément, il existe Δ_δ système de transition d'un automate dans $\mathcal{P}_k\text{-AP}^{\mathcal{D}_k(\vec{C})}(A_1, \ldots, A_k)$ tel que pour tout $\omega \in k$-Pile, $\delta^{\vec{C}} \leq_{\mathcal{D}_k,\omega} {\Delta_\delta}^{\mathcal{D}_k(\vec{C})}$.*

Preuve : Définissons de système de transition (p, α, q)-standard Δ_δ par :

$$\begin{aligned}\Delta_\delta &= \{(s, \varepsilon, a_k \cdots a_1, \vec{o'}, \mathrm{prod}_{\bar{a_i}}, s) \mid \forall j \in [1,k], a_j \in A_i^{\mathcal{C}},\ a_i \neq b_i, \vec{o'} \in \{0,1\}^m\} \\ &\cup \{(s, \varepsilon, a_k \cdots a_1, \vec{o'}, \mathrm{prod}_{\bar{a_i}}, q) \mid \forall j \in [1,k], a_j \in A_i^{\mathcal{C}},\ a_i = b_i, \vec{o'} \in \{0,1\}^m\} \\ &\cup \Delta_{\delta, b_k \cdots b_1},\end{aligned}$$

et

- si $b_k \cdots b_1 \neq \perp^k$, alors

$$\Delta_{\delta, b_k \cdots b_1} = \{(p, \alpha, a_k \cdots a_2 b_1, \vec{o}, \mathrm{stay}, q) \mid \forall j \in [2,k],\ a_j \in A_j^{\mathcal{C}}\},$$

- si $b_j \cdots b_1 = \perp^j$ et $b_{j+1} \neq \perp$ alors

$$\Delta_{\delta, b_k \cdots b_1} = \{(p, \alpha, a_k \cdots a_{j+1} \perp \cdots \perp, \vec{o}, \mathrm{stay}, q) \mid \forall \ell \in [2,k],\ a_\ell \in A_\ell^{\mathcal{C}}\}.$$

D'après le lemme précédent, il est clair que pour tout $\omega \in k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$, $\mathrm{pop}_i(\omega)$ est défini ssi il existe $u_1 \in \widehat{A_{1,i-1}}^*$ et $a \in A_i$ tels que $\mathrm{mot}_i(\mathcal{D}_k(\omega)) \in A_i^{\mathcal{C}*} \cdot a \cdot u_1$. Donc, d'après le lemme 9.3.4 et la définition de l'instruction prod,

$$(p, \alpha, \omega) \rightarrow_{\delta^{\vec{C}}} (q, \varepsilon, \mathrm{pop}_i(\omega)) \text{ ssi } (p, \alpha, \mathcal{D}_k(\omega)) \rightarrow^*_{\Delta_\delta{}^{\mathcal{D}(\vec{C})}} (q, \varepsilon, \mathcal{C}_k(\varphi_k(\omega) \bullet \overline{u_1} \bullet \overline{a})).$$

Or, toujours d'après le lemme 9.3.4,

$$\varphi_k(\omega) \bullet \overline{u_1} \bullet \overline{a} = \varphi_k(\mathrm{pop}_i(\omega)) = \varphi_k(\mathrm{pop}_i(\omega)).$$

En substituant $\varphi_k(\omega) \bullet \overline{u_1} \bullet \overline{a}$ par $\varphi_k(\mathrm{pop}_i(\omega))$ dans le calcul ci-dessus, nous obtenons donc, pour tout $\omega \in k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$,

$$(p, \alpha, \omega) \vdash_{\delta^{\vec{C}}} (q, \varepsilon, \mathrm{pop}_i(\omega)) \text{ ssi } (p, \alpha, \mathcal{D}_k(\omega)) \vdash^*_{\Delta_\delta{}^{\mathcal{D}(\vec{C})}} (q, \varepsilon, \mathcal{D}_k(\mathrm{pop}_i(\omega))).$$

□

Proposition 9.3.6. *Pour tous alphabets de piles $A_1, \ldots, A_k$, il existe des alphabets de piles $B_1, \ldots B_k$ tel que pour tout contrôleur $\vec{C} \in i\text{-}\mathsf{Pile}(A_1, \ldots, A_i)$, il existe un vecteur $\vec{R}$ d'ensembles dans $\mathcal{P}_i(B_1, \ldots, B_i)$ tel que*

$$k\text{-}\mathsf{AP}^{[\vec{C}]_i}(A_1, \ldots, A_k) \sqsubseteq \mathcal{P}_k\text{-}\mathsf{AP}^{(\vec{R})_i}(B_1, \ldots, B_k)$$

et

- *$\vec{C}$ et $\vec{R}$ sont de même taille,*
- *si $\vec{C} \in \mathsf{PREC}_i(A_1, \ldots, A_i)$, alors $\vec{R} \in \mathsf{REC}_i(B_1, \ldots, B_i)$.*

Preuve : En utilisant les lemmes 9.3.3 et 9.3.5, il est clair que pour tous $B_1, \ldots, B_k$, avec $|B_2| = \cdots = |B_k| = 1$, et $\vec{D} \in k\text{-}\mathsf{Pile}(B_1, \ldots, B_k)$,

$$k\text{-}\mathsf{NAP}^{\vec{D}}(B_1, \ldots, B_k) \sqsubseteq_{\mathcal{D}_k} \mathcal{P}_k\text{-}\mathsf{AP}^{\mathcal{D}_k(\vec{D})}(B_1, \ldots, B_k).$$

D'après la remarque 9.1.3(2), pour tout $u \in \mathcal{P}_k$, $\mathrm{p}_i(\mathcal{C}_k(u)) = \mathcal{C}_i(\mathrm{mot}_i(\mathcal{C}_k(u)))$. Puisque $\mathcal{D}_k = \varphi_k \circ \mathcal{C}_k$, pour tout $S \subseteq i\text{-}\mathsf{Pile}$, $1 \leq i \leq k$, pour tout $\omega \in k\text{-}\mathsf{Pile}(B_1, \ldots, B_k)$, $\mathrm{p}_i(\omega) \in$

S ssi $\mathrm{mot}_i(\mathcal{D}_k(\omega)) \in \varphi_i(S)$. Nous avons donc pour tout vecteur $\vec{D}$ de sous-ensembles de i-$\mathsf{Pile}(A_1, \ldots, A_k)$,

$$k\text{-}\mathsf{NAP}^{[\vec{D}]_i}(B_1, \ldots, B_k) \sqsubseteq \mathcal{P}_k\text{-}\mathsf{AP}^{(\varphi_i(\vec{D}))_i}(B_1, \ldots, B_k) \tag{12}$$

En particulier, d'après le théorème 6.2.7,

$$\text{si } \vec{D} \in \mathsf{PREC}_i(B_1, \ldots, B_k), \text{ alors } \varphi_i(\vec{D}) \in \mathsf{REC}_i(B_1, \ldots, B_k).$$

Enfin, nous avons montré précédemment (proposition 8.2.3) que pour tous alphabets de piles $A_1, \ldots, A_k$ il existe des alphabets de piles $B_1, \ldots, B_k$, avec $|B_2| = \cdots = |B_k| = 1$ tels que pour tous $C_1, \ldots, C_m \in i$-$\mathsf{Pile}(A_1, \ldots, A_i)$, il existe des ensembles $D_1, \ldots D_m \in i$-$\mathsf{Pile}(B_1, \ldots, B_i)$ tels que

$$k\text{-}\mathsf{AP}^{[\vec{C}]_i}(A_1, \ldots, A_k) \sqsubseteq k\text{-}\mathsf{NAP}^{[\vec{D}]_i}(B_1, \ldots, B_k).$$

De plus, si $\vec{C} \in \mathsf{PREC}_i(A_1, \ldots, A_i)$, alors $\vec{D} \in \mathsf{PREC}_i(B_1, \ldots, B_i)$. En appliquant (12) nous avons donc

$$k\text{-}\mathsf{AP}^{[\vec{C}]_i}(A_1, \ldots, A_k) \sqsubseteq k\text{-}\mathsf{NAP}^{[\vec{D}]_i}(B_1, \ldots, B_k) \sqsubseteq \mathcal{P}_k\text{-}\mathsf{AP}^{(\varphi_i(\vec{D}))_i}(B_1, \ldots, B_k),$$

ce qui prouve la proposition.

□

9.3.2 Contrôles par des propriétés SOM-définissables

Théorème 9.3.7. *Pour tout $k \geq 1$,*

$$k\text{-}\mathsf{AP}^{\mathsf{PREC}_k} \sqsubseteq k\text{-}\mathsf{AP}^{\vec{\emptyset}}.$$

En d'autres mots, pour tous alphabets de piles $A_1, \ldots, A_k$, pour tout vecteur $\vec{C}$ de langages dans $\mathsf{PREC}_k(A_1, \ldots, A_k)$, il existe des alphabets $B_1, \ldots B_k$ tels que $k\text{-}\mathsf{AP}^{\vec{C}}(A_1, \ldots, A_k) \sqsubseteq k\text{-}\mathsf{AP}^{\vec{\emptyset}}(B_1, \ldots, B_k)$.

Preuve : Nous prouvons par induction sur $i \in [0, k]$ que

$$k\text{-}\mathsf{AP}^{[\mathsf{PREC}_i]_i} \sqsubseteq k\text{-}\mathsf{AP}^{\vec{\emptyset}}.$$

Base : si $i = 0$, alors $\mathsf{PREC}_0 = \{\{\bot_0\}, \emptyset\}$, la proposition est évidente.
Pas d'induction : supposons la propriété vraie pour $i \geq 0$, alors d'après les propositions 9.3.6 et 9.2.10 et l'hypothèse d'induction :

$$k\text{-}\mathsf{AP}^{[\mathsf{PREC}_{i+1}]_{i+1}} \sqsubseteq \mathcal{P}_k\text{-}\mathsf{AP}^{(\mathsf{REC}_{i+1})_{i+1}} \sqsubseteq k\text{-}\mathsf{AP}^{[\mathsf{PREC}_i]_i} \sqsubseteq k\text{-}\mathsf{AP}^{\vec{\emptyset}}.$$

La propriété est donc vraie pour tout i, en particulier, pour $i = k$, nous avons

$$k\text{-}\mathsf{AP}^{\mathsf{PREC}_k} = k\text{-}\mathsf{AP}^{[\mathsf{PREC}_k]_k} \sqsubseteq k\text{-}\mathsf{AP}^{\vec{\emptyset}}.$$

□

9.3.3 Automates à k-piles "symétriques"

La classe des automates à k-piles utilisant les instructions $\text{push}_{i,a}$ et $\overline{\text{push}}_{i,a}$ est équivalente par simulation déterministe à la classe k-AP.

Théorème 9.3.8. *Pour tout $k \geq 1$*

$$k\text{-AP}^{\vec{\emptyset}} \equiv \overline{k\text{-AP}}^{\vec{\emptyset}}.$$

Preuve : En composant les proposition 9.3.6 et 9.2.8, nous obtenons $k\text{-AP}^{\vec{\emptyset}} \sqsubseteq \mathcal{P}_k\text{-AP}^{\vec{\emptyset}} \sqsubseteq \overline{k\text{-AP}}^{\vec{\emptyset}}$.

Inversement, nous avons remarqué précédemment (voir §7.2.3) que $\overline{k\text{-AP}}^{\vec{\emptyset}} \sqsubseteq k\text{-AP}^{\mathsf{PREC}_k}$ d'où, en appliquant le théorème 9.3.7 : $k\text{-AP}^{\vec{\emptyset}} \equiv \overline{k\text{-AP}}^{\vec{\emptyset}}$

□

9.3.4 Contrôles par un automate à ℓ-piles

Définition 9.3.9. *Nous notons $\overrightarrow{\mathsf{DLANG}}_k(\Sigma)$ l'ensemble des vecteurs $\vec{L} = (L_1, \ldots, L_m)$ de langages dans Σ^*, tels qu'il existe m automates déterministes*

$$\mathcal{A}_i = (Q, \Sigma, (A_1, \ldots, A_k), \Delta, q_0, \bot, F_i) \in k\text{-AP}^{\vec{\emptyset}}, \text{ pour } i \in [1, m]$$

tels que pour tout $i \in [1, m]$, $\mathrm{L}(\mathcal{A}_i) = L_i$.

Lemme 9.3.10. *Tout vecteur $\vec{R}$ de langages dans REC_{k+1}, $k \geq 0$, appartient à $\overrightarrow{\mathsf{DLANG}}_k$.*

Preuve : D'après la proposition 9.1.9, il existe m automates déterministes

$$\mathcal{A}_i = (Q, \Sigma, (A_1, \ldots, A_k), (\vec{R})_k, \Delta, q_0, \bot, F_i) \in \mathcal{P}_k\text{-AP}^{(\mathsf{REC}_k)_k}, \text{ pour } i \in [1, m]$$

tels que pour tout $i \in [1, m]$, $\mathrm{L}(\mathcal{A}_i) = L_i$. En appliquant la proposition 9.2.5 puis le théorème 9.3.7, il existe donc m automates déterministes

$$\mathfrak{B}_i = (Q', \Sigma, (B_1, \ldots, B_k), \Delta', q_0, \bot, F_i) \in k\text{-AP}^{\vec{\emptyset}}, \text{ pour } i \in [1, m]$$

tels que pour tout $i \in [1, m]$, $\mathrm{L}(\mathcal{B}_i) = L_i$.

□

Théorème 9.3.11. *Pour tous $k, \ell \geq 1$, $(k+\ell)\text{-AP}^{\vec{\emptyset}} \equiv k\text{-AP}^{(\overrightarrow{\mathsf{DLANG}}_\ell)_1}$.*

En composant les proposition 9.3.6 et 9.2.12, nous obtenons

$$(k+\ell)\text{-AP}^{\vec{\emptyset}} \equiv k\text{-AP}^{(\mathsf{REC}_{\ell+1})_1}.$$

En appliquanquant ensuite le le lemme 9.3.10, nous avons donc $(k+\ell)\text{-AP}^{\vec{\emptyset}} \sqsubseteq k\text{-AP}^{(\overrightarrow{\mathsf{DLANG}}_\ell)_1}$. L'inclusion inverse est prouvée dans le lemme suivant :

Lemme 9.3.12. *Pour tous $k, \ell \geq 1$, $k\text{-AP}^{(\overrightarrow{\mathsf{DLANG}}_\ell)_1} \sqsubseteq (k+\ell)\text{-AP}^{\emptyset}$.*

La simulation utilisée pour la preuve de ce lemme est construite sur le même modèle que la simulation id_η utilisée dans la preuve de la proposition 9.2.10, mais elle augmente en plus le niveau de la pile.

Considérons deux applications : $\eta_1 : {A_{\ell+1}}^* \to B$ où B est un alphabet quelconque et $\eta_2 : {A_{\ell+1}}^* \to l\text{-}\mathsf{Pile}^{\perp'}(A_1, \ldots, A_\ell)$ telles que $\eta_2(\varepsilon) = \perp'_\ell$ et $\perp' = (\perp, \eta_1(\varepsilon))$.

Nous définissons pour tout $k \geq 0$, l'application

$$\mathrm{id}_{\eta_1,\eta_2} : k\text{-}\mathsf{Pile}^{\perp}(A_{\ell+1}, \ldots, A_{k+\ell}) \to k + \ell\text{-}\mathsf{Pile}^{\perp'}(A_1, \ldots, A_\ell, A_{\ell+1} \times B, A_{\ell+2}, \ldots A_{\ell+k})$$

vérifiant pour tout $\omega \in k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$:

- la pile tronquée $\mathrm{id}_{\eta_1,\eta_2}(\omega)^{-\ell}$ est égale à $\mathrm{id}_{\eta_1}(\omega)$ (voir §9.2.4, figure 21),
- la projection au niveau ℓ, $\mathrm{p}_\ell(\mathrm{id}_{\eta_1,\eta_2}(\omega))$ est égale à $\eta_2(\mathrm{mot}_1(\omega))$,
- ces propriétés se propagent à tout "préfixe" de $\mathrm{id}_{\eta_1,\eta_2}(\omega)$: pour tout $i \in [\ell+1, \ell+k]$, $\mathrm{pop}_i(\mathrm{id}_{\eta_1,\eta_2}(\omega)) = \mathrm{id}_{\eta_1,\eta_2}(\mathrm{pop}_i(\omega))$.

La figure 23 illustre cette transformation.

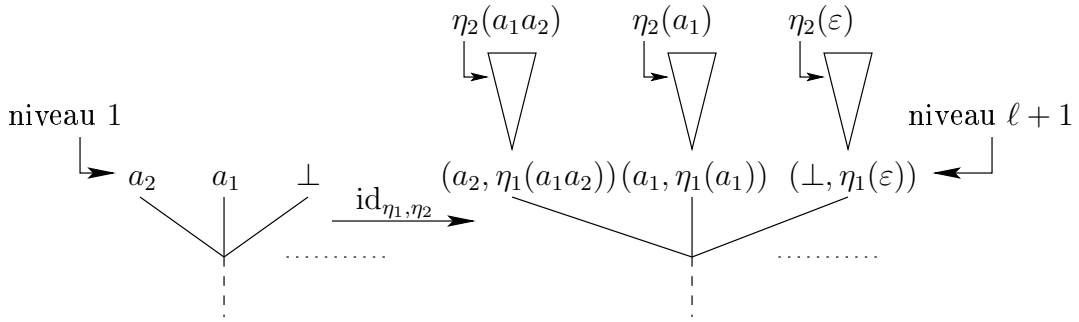


FIG. 23 – Application de la fonction $\mathrm{id}_{\eta_1,\eta_2}$

Une définition récursive de $\mathrm{id}_{\eta_1,\eta_2}$ est la suivante :

nous posons par convention $\mathrm{id}_{\eta_1,\eta_2}(\varepsilon) = \varepsilon$, alors pour tout $\omega = a[\omega_1]\omega' \in k\text{-}\mathsf{Pile}$:

- $\mathrm{id}_{\eta_1,\eta_2}(\omega) = (a, \eta_1(\mathrm{mot}_1(\omega)))[\eta_2(\mathrm{mot}_1(\omega))]\,\mathrm{id}_{\eta_1,\eta_2}(\omega')$, si $k = 1$,
- $\mathrm{id}_{\eta_1,\eta_2}(\omega) = a[\mathrm{id}_{\eta_1,\eta_2}(\omega_1)]\,\mathrm{id}_{\eta_1,\eta_2}(\omega')$ sinon.

Intuitivement, pour simuler un k-AP dont les contrôleurs sont calculés par un ℓ-AP déterministe et sans ε-transitions, $(\eta_1(u), \eta_2(u))$ représente **l'état total** atteint par l'automate contrôleur à la fin de la lecture de u. Si l'automate effectue des ε-transitions, ce couple n'est pas unique, il faut donc déterminer comment le choisir. De plus la possibilité qu'il existe dans le graphe de l'automate des suites infinies de ε-transitions oblige à calculer par avance la valeur du vecteur caractéristique de u qui apparaîtra dans $\eta_1(u)$, en plus de l'état atteint.

Preuve : Nous procédons en deux étapes : nous construisons tout d'abord un automate déterministe de niveau ℓ reconnaissant les oracles et près à être composé avec l'automate contrôlé, puis nous construisons le $(k+\ell)$-AP simulant l'automate contrôlé.

Première étape : soit $\vec{L} = (L_1, \ldots, L_m)$ un vecteur de langages dans ${A_{\ell+1}}^*$. Supposons qu'il existe un automate *déterministe* $\mathcal{A} = (Q, A_{\ell+1}, (B_1, \ldots, B_\ell), \Delta, p_0, \perp', Q) \in \ell\text{-}\mathsf{AP}^{\vec{\emptyset}}$ et $F_1, \ldots, F_m$ des sous-ensembles de Q tels que pour tout $i \in [1, m]$, l'automate $\mathcal{A}_i$ obtenu en restreignant les état finaux de $\mathcal{A}$ à F_i reconnaît le langage L_i.

Nous pouvons sans perte de généralité supposer que $\mathcal{A}$ est "complet" au sens suivant : pour chaque sommet du graphe de calcul de $\mathcal{A}$,

- soit il existe un unique arc sortant, et cet arc est étiqueté par ε
- soit pour chaque lettre de $A_{\ell+1}$, il existe exactement un arc sortant étiqueté par cette lettre

(Si l'automate n'est pas de cette forme, il suffit d'ajouter un état et ainsi que les arcs manquants se dirigeant vers cet état.)

Nous procédons à nouveau en deux étapes. Tout d'abord nous construisons un nouvel ℓ-AP déterministe et contrôlé par des propriétés définissables en LSOM, admettant un calcul pour chaque $u \in {A_{\ell+1}}^*$, et capable d'indiquer la valeur de $\chi_{\vec{L}}(u)$. Dans un second temps, nous supprimons les contrôles de cet automate, tout en conservant ces propriétés.

Sous-étape 1 : définissons tout d'abord les contrôleurs que nous allons utiliser.

- un vecteur $\vec{C}$ composé de tous les $C_{p,\vec{o}}$ pour $p \in Q$ et $\vec{o} \in \{0,1\}^m$ tels que :
  $C_{p,\vec{o}}$ contient exactement tous les ω tels que pour tout $i \in [1,m]$,

$$[\exists p' \in F_i, \omega',\ (p,\varepsilon,\omega)\vdash_{\mathcal{A}}^* (p',\varepsilon,\omega')] \text{ ssi } o_i = 1,$$

- un vecteur $\vec{D}$ composé de tous les D_p, pour $p \in Q$ tels que
  D_p contient exactement tous les ω tels que

$$\exists p' \in Q, \omega', a \in A_{\ell+1},\ (p,a,\omega)\vdash_{\mathcal{A}}^* (p',\varepsilon,\omega').$$

Ces vecteurs sont tous les deux clairement SOM-définissables dans la structure $\mathbf{Pile_\ell}$ et appartiennent donc à PREC_ℓ. De plus, il est évident que tout $ua \in {A_{\ell+1}}^*$, si il existe un calcul de la forme

$$(p_0, ua, \bot'_\ell)\vdash_{\mathcal{A}}^* (p,a,\omega)\vdash_{\mathcal{A}} (p',\varepsilon,\omega')$$

alors :

- $\omega' \in C_{p',\chi_{\vec{L}}(ua)}$,
- $\omega' \in D_{q'}$ ssi ce calcul conduit à la lecture en entrée d'au moins encore une lettre (i.e., le reste du calcul n'est pas une suite infinie de ε-transitions).

Posons $\widetilde{Q} = (Q \cup \{s_\bot\}) \times \{0,1\}^m$ (où $s_\bot$ est un nouveau symbole). Nous définissons deux applications $\eta_1 : {A_{\ell+1}}^* \to \widetilde{Q}_1$ et $\widetilde{\eta}_2 : {A_{\ell+1}}^* \to \ell\text{-}\mathsf{Pile}^{\bot'}(B_1,\ldots,B_\ell)$ de la façon suivante :

- $\eta_1(\varepsilon) = (p_0, \chi_{\vec{L}}(\varepsilon))$ et $\widetilde{\eta}_2(\varepsilon) = \bot'_\ell$
- pour tout $ua \in {A_{\ell+1}}^*$,
  - si il existe un calcul
$$(p_0, ua, \bot'_\ell)\vdash_{\mathcal{A}}^* (p,a,\omega)\vdash_{\mathcal{A}} (p',\varepsilon,\omega')$$
    alors $\eta_1(ua) = (p', \chi_{\vec{L}}(ua))$ et $\widetilde{\eta}_2(ua) = \omega'$,
  - sinon, $\eta_1(ua) = (s_\bot, (0,\ldots,0))$ (remarquons que dans ce cas $(0,\ldots,0) = \chi_{\vec{L}}(ua)$)) et $\widetilde{\eta}_2(ua) = \widetilde{\eta}_2(u)$.

En utilisant les contrôleurs $\vec{C}$ et $\vec{D}$, nous allons construire un nouvel automate déterministe $\mathcal{A}_1$ dont l'ensemble des états contient $\widetilde{Q}$ et vérifiant la propriété $\mathbf{P_1}$ suivante :

$$\forall u \in {A_{\ell+1}}^*, \textit{ il existe un unique couple } (p,\vec{o}) \in \widetilde{Q}, \textit{ il existe } \omega \in \ell\text{-}\mathsf{Pile} \textit{ tels que}$$

$$(\eta_1(\varepsilon), u, \bot'_\ell)\vdash_{\mathcal{A}'}^* ((p,\vec{o}),\varepsilon,\omega')$$

$$\textit{de plus, } \eta_1(u) = (p,\vec{o}) \textit{ et } \widetilde{\eta}_2(u) = \omega'.$$

Construisons cet automate. Nous posons

$$\mathcal{A}_1 = (Q_1 \cup \widetilde{Q}, A_{\ell+1}, (B_1, \ldots, B_\ell), \Delta_1, p'_0, \perp', Q) \in \ell\text{-}\mathsf{AP}^{(\vec{C},\vec{D})},$$

où $p'_0 = (p_0, \chi_{\vec{L}}(\varepsilon))$ (d'après la remarque 10.1.13, il existe une procédure effective permettant de calculer la valeur de $\chi_{\vec{L}}(\varepsilon)$), $Q_1 = Q \cup \{p_l \mid p \in Q\}$ (les nouveaux états p_l permettront de signaler que l'automate vient d'effectuer une lecture sur la bande d'entrée) et Δ_1 est obtenu de Δ de la façon suivante :

- $\forall (p, a, w, \text{instr}, p') \in \Delta$, $\forall \vec{c} \in \{0,1\}^{|\vec{C}|}$, $\forall \vec{d} \in \{0,1\}^{|\vec{D}|}$,

$$1.\ (p, a, w, (\vec{c}, \vec{d}), \text{instr}, p'_l) \in \Delta_1,$$

- $\forall (p, \varepsilon, w, \text{instr}, p') \in \Delta$, $\forall \vec{c} \in \{0,1\}^{|\vec{C}|}$, $\forall \vec{d} \in \{0,1\}^{|\vec{D}|}$,

$$2.\ (p, a, w, (\vec{c}, \vec{d}), \text{instr}, p') \in \Delta_1,$$

- $\forall (p, \vec{o}) \in \widetilde{Q}$, $\forall \vec{c} \in \{0,1\}^{|\vec{C}|}$ tel que $c_{p,\vec{o}} = 1$, $\forall \vec{d} \in \{0,1\}^{|\vec{D}|}$, $\forall w$,

$$3.\ (p_l, \varepsilon, w, (\vec{c}, \vec{d}), \text{stay}, (p, \vec{o})) \in \Delta_1,$$

- $\forall (p, \vec{o})$, $\forall \vec{c} \in \{0,1\}^{|\vec{C}|}$, $\forall \vec{d} \in \{0,1\}^{|\vec{D}|}$, $\forall w$,
  - si $p \neq s_\perp$ et $d_p = 1$, alors

$$4.1.\ ((p, \vec{o}), \varepsilon, w, (\vec{c}, \vec{d}), \text{stay}, p) \in \Delta_1,$$

  - si $p = s_\perp$ ou $d_p = 0$, alors $\forall a \in A_{\ell+1}$

$$4.2.\ ((p, \vec{o}), a, w, (\vec{c}, \vec{d}), \text{stay}, (s_\perp, (0, \ldots, 0))) \in \Delta_1.$$

Vérifions la propriété suivante : $\forall ua \in {A_{\ell+1}}^*$,

$$(\eta_1(u), a, \widetilde{\eta}_2(u)) \vdash^*_{\mathcal{A}_1} (\eta_1(ua), \varepsilon, \widetilde{\eta}_2(ua)). \tag{13}$$

Supposons $\eta_1(u) = (p, \vec{o})$,

- si ua n'admet pas de calcul dans $\mathcal{A}$, alors soit il n'existe pas non plus de calcul de u et dans ce cas $p = s_\perp$, soit il existe un calcul de u aboutissant en $(p, \widetilde{\eta}_2(u))$ et il n'y a ensuite plus que des ε-transitions applicables à l'infini, i.e., $\widetilde{\eta}_2(u) \notin D_p$.
  Alors, en appliquant la transition (4.2), nous obtenons bien

$$(\eta_1(u), a, \widetilde{\eta}_2(u)) \vdash^*_{\mathcal{A}_1} (\eta_1(ua), \varepsilon, \widetilde{\eta}_2(ua)),$$

- sinon, il existe dans $\mathcal{A}$ un calcul de u aboutissant en $(p, \widetilde{\eta}_2(u))$ et qui continue pour lire a. En appliquant la transition (4.1), puis les transitions (2), et pour finir la transition (1) correspondant à la lecture de a, nous obtenons

$$(\eta_1(u), a, \widetilde{\eta}_2(u)) \vdash^*_{\mathcal{A}_1} (p', \varepsilon, \omega'),$$

  où (p', ω') est le premier état total rencontré lors du calcul de ua par $\mathcal{A}$. En appliquant maintenant une transition (3), nous obtenons bien

$$(\eta_1(u), a, \widetilde{\eta}_2(u)) \vdash^*_{\mathcal{A}_1} (\eta_1(ua), \varepsilon, \widetilde{\eta}_2(ua)).$$

Une simple induction utilisant les calculs (13) permet de vérifier que cet automate satisfait bien la propriété $\mathbf{P_1}$ énoncée précédemment.

Sous-étape 2 : puisque les contrôleurs de l'automate $\mathcal{A}_1$ appartiennent à PREC_ℓ, le théorème 9.3.7 assure de l'existence d'une fonction $\mathcal{S} : \ell\text{-}\mathsf{Pile}(B_1, \ldots, B_\ell) \to \ell\text{-}\mathsf{Pile}(A_1, \ldots, A_\ell)$ définie pour toute pile accessible par un calcul dans $\mathcal{A}_1$, et d'un automate déterministe $\mathcal{A}_2 \in \ell\text{-}\mathsf{AP}^{\vec{\emptyset}}(A_1, \ldots, A_\ell)$ tel que $\mathcal{A}_1 \leq_{\mathcal{S}} \mathcal{A}_2$. Posons $\eta_2 = \widetilde{\eta}_2 \circ \mathcal{S}$. L'ensemble d'états de ce nouvel automate contient donc $\tilde{Q}$, et d'après $\mathbf{P_1}$ et les propriétés de simulation, il est clair que $\mathcal{A}_2$ vérifie la propriété $\mathbf{P_2}$ suivante :

$$\forall u \in A_{\ell+1}, \textit{ il existe un unique couple } (p, \vec{o}) \in \widetilde{Q}, \textit{ il existe } \omega \in \ell\text{-}\mathsf{Pile} \textit{ tels que}$$

$$(\eta_1(\varepsilon), u, \perp'_\ell) \vdash^*_{\mathcal{A}'} ((p, \vec{o}), \varepsilon, \omega')$$

$$\textit{de plus, } \eta_1(u) = (p, \vec{o}) \textit{ (et donc } \chi_{\vec{L}}(u) = \vec{o}) \textit{ et } \eta_2(u) = \omega'.$$

Seconde étape : soit $\mathcal{B} = (Q', \Sigma, (A_{1+\ell}, \ldots, A_{k+\ell}), (\vec{L})_1, \Delta', q_0, F')$ un automate de niveau k. Nous prouvons maintenant qu'il existe $\mathcal{B}' \in (\ell + k)\text{-}\mathsf{AP}^{\vec{\emptyset}}(A_1, \ldots, A_\ell, A'_{\ell+1}, A_{\ell+2}, \ldots, A_{k+\ell})$ (avec $A'_{\ell+1} = A_{\ell+1} \times \tilde{Q}$) tel que $\mathcal{B} \leq_{\mathrm{id}_{\eta_1,\eta_2}} \mathcal{B}'$.

Nous posons $\perp' = (\perp, p'_0)$, $Q'' = Q' \cup \{q_\delta \mid q \in Q_1 \cup \widetilde{Q}\}$ et

$$\mathcal{B}' = (Q'', \Sigma, (A_1, \ldots, A_\ell, B_{\ell+1}, A_{\ell+2}, \ldots, A_{k+\ell}), \Delta'', q_0, F')$$

où Δ'' est composé de l'union, pour $\delta \in \Delta_2$ des systèmes Δ''_δ définis de la façon suivante : (dans tout ce qui suit, si $a \in A_i \cup \{\perp\}$, $i \in [\ell+1, \ell+k]$, alors a' désigne le symbole a si $a \neq \perp$ et $\perp'$ sinon)

1. si $\delta = (q, \alpha, a_{k+\ell} \cdots a_{\ell+1}, \vec{o}, \mathrm{instr}, q')$ et $\mathrm{instr} \neq \mathrm{push}_{1,a}$, alors Δ''_δ est l'ensemble pour $i \in [1, \ell]$, $a_i \in A_i$ et $p \in Q \cup \{s_\perp\}$ des transitions
$$(q, \alpha, a'_{k+\ell} \cdots a'_{\ell+2}(a_{\ell+1} a_\ell \cdots a_1, (p, \vec{o})), \mathrm{instr}^{+\ell}, q'),$$
2. si $\delta = (q, \alpha, a_{k+\ell} \cdots a_{\ell+1}, \vec{o}, \mathrm{push}_{1,a}, q')$ alors
   - $\forall a_i \in A_i$, $i \in [1, \ell]$, $\forall p \in Q \cup \{s_\perp\}$,
   $$(q, \alpha, a'_{k+\ell} \cdots a'_{\ell+2}(a_{\ell+1}, (p, \vec{o}))a_\ell \cdots a_1, \mathrm{push}_{1,(a,(p,\vec{o}))}, (p, \vec{o})_\delta) \in \Delta''_\delta,$$
   - $\forall (p', a, a_\ell \cdots a_1, \mathrm{instr}, p'')$, $p'' \notin \widetilde{Q}$, $a \in A_{\ell+1} \cup \{\varepsilon\}$
   $$({p'}_\delta, \varepsilon, a'_{k+\ell} \cdots a'_{\ell+2}(a, (p, \vec{o}))a_\ell \cdots a_1, \mathrm{instr}, {p''}_\delta),$$
   - $\forall (p', a, a_\ell \cdots a_1, \mathrm{instr}, (p'', \vec{o}))$, $a \in A_{\ell+1} \cup \{\varepsilon\}$
   $$({p'}_\delta, \varepsilon, a'_{k+\ell} \cdots a'_{\ell+2}(a, (p, \vec{o}))a_\ell \cdots a_1, \mathrm{instr\,change}_{\ell+1,(a,(p'',\vec{o}'))}, q').$$

Vérifions maintenant de pour chaque $\delta \in \Delta'$, pour chaque $\omega \in k\text{-}\mathsf{Pile}$, $\delta^{(\vec{L})_1} \leq_{\mathrm{id}_{\eta_1,\eta_2},\omega} \Delta''_\delta$.

- Si δ ne contient pas d'instruction push au niveau 1, alors puisque $\mathrm{top}_{\ell+1}(\mathrm{id}_{\eta_1,\eta_2}(\omega) = (\mathrm{top}_1(\omega), \eta_1(\mathrm{mot}_1(\omega))$ et la seconde composante de $\eta_1(\mathrm{mot}_1(\omega))$ est $\chi_{\vec{L}}(\mathrm{mot}_1(\omega))$, il est évident que δ est applicable à ω ssi Δ''_δ est applicable à $\mathrm{id}_{\eta_1,\eta_2}(\omega)$. De plus, une induction évidente sur $k \geq 1$ permet de vérifier que pour tout $i \in [1, k]$
  - $\mathrm{id}_{\eta_1,\eta_2}(\mathrm{pop}_i(\omega)) = \mathrm{pop}_{\ell+i}(\mathrm{id}_{\eta_1,\eta_2}(\omega))$,

– si $i \neq 1$, $\mathrm{id}_{\eta_1,\eta_2}(\mathrm{push}_{i,a}(\omega)) = \mathrm{push}_{i+\ell,a}(\mathrm{id}_{\eta_1,\eta_2}(\omega))$.

Nous avons donc clairement, $\delta^{(\vec{L})_1} \leq_{\mathrm{id}_{\eta_1,\eta_2},\omega} \Delta''_\delta$.

– Si $\delta = (q, \alpha, w, \vec{\sigma}, \mathrm{push}_{1,a}, q')$, le système Δ''_δ est clairement (q, α, q')-standard, et par le même argument que pour le point précédent, δ est applicable à ω ssi Δ''_δ est applicable à $\mathrm{id}_{\eta_1,\eta_2}(\omega)$, et une simple induction permet de vérifier que

$$(q, \alpha, \omega) \vdash_{\delta^{(\vec{L})_1}} (q', \varepsilon, \omega') \text{ ssi } (q, \alpha, \mathrm{id}_{\eta_1,\eta_2}(\omega)) \vdash^*_{\Delta''_\delta} (q', \varepsilon, \mathrm{id}_{\eta_1,\eta_2}(\omega')).$$

□

La classe des langages reconnus par $(k+1)$-AP (déterministe) non contrôlés est donc exactement la classe des langages reconnus par 1-AP (déterministe) contrôlés par un vecteur dans $\overrightarrow{\mathsf{DLANG}_k}$.

Corollaire 9.3.13. *Pour tout* $k \geq 0$, $\mathsf{DLANG}_{k+1} = \mathsf{DLANG}_1^{(\overrightarrow{\mathsf{DLANG}_k})_1}$.

Chapitre 10

Applications

10.1 Théorie des langages

10.1.1 Langages de niveau k versus langages k-réguliers

Proposition 10.1.1. *Tout langage dans* REC_{k+1}, $k \geq 0$ *est reconnu par un* k*-AP non contrôlé et déterministe.*

Preuve : Nous avons montré (proposition 9.1.8) que tout langage dans REC_k était reconnu par un automate déterministe dans $\mathcal{P}_k$-$\mathsf{AP}^{\vec{\emptyset}}$. Or, d'après la proposition 9.2.5, $\mathcal{P}_k$-$\mathsf{AP}^{\vec{\emptyset}} \sqsubseteq k$-$\mathsf{AP}^{\vec{\emptyset}}$. La simulation déterministe préservant la classe des langages reconnus par automate déterministe, tout langage dans REC_k est reconnu par un automate déterministe dans $\mathcal{P}_k$-$\mathsf{AP}^{\vec{\emptyset}}$.

□

Proposition 10.1.2. *Pour* $R \in \mathsf{REC}_k$ *fixé et* $u \in \mathcal{P}_k$ *de longueur* n*, le problème "*$u \in R$*?" est décidable en temps* $\mathcal{O}(n)$.

Preuve : D'après la preuve de la proposition 9.1.8, R est reconnu par un $\mathcal{P}_k - AP$ déterministe sans ε-transitions. Chaque transition d'un $\mathcal{P}_k - AP$ correspond à une suite déterministe d'au plus k transitions dans un $k - AP$ utilisant les instructions classiques. On peut donc tester que $u \in R$ en nk opérations.

□

Nous montrons maintenant que la projection de REC_{k+1} dans ${A_{k+1}}^*$ forme exactement l'ensemble des langages dans ${A_{k+1}}^*$ reconnus par automates de niveau k. Ceci permet de donner une nouvelle caractérisation de LANG_k.

Théorème 10.1.3. *Pour tout* $k \geq 1$, $\pi_{A_{k+1}}(\mathsf{REC}_{k+1}) = \mathsf{LANG}_k(A_{k+1})$.

Nous prouvons dans les deux lemmes suivants les inclusions permettant de démontrer le théorème.

Lemme 10.1.4. *Pour tout* $k \geq 1$, $\pi_{A_{k+1}}(\mathsf{REC}_{k+1}) \subseteq \mathsf{LANG}_k(A_{k+1})$.

Preuve : Soit $R \in \mathsf{REC}_{k+1}$, d'après le théorème 10.1.1, $R \in \mathsf{LANG}_k$. Soit $\mathcal{A}$ un k-$\mathsf{AP}^{\emptyset}$ reconnaissant R, on obtient un nouvel automate dans k-$\mathsf{AP}^{\vec{\emptyset}}$ reconnaissant $\pi_{A_{k+1}}$ en substituant dans $\mathcal{A}$ chaque transition $(p, \alpha, w, \text{instr}, q)$ où $\alpha \notin A_{k+1}$ par $(p, \varepsilon, w, \text{instr}, q)$.

□

Lemme 10.1.5. *Pour tout $k \geq 1$, $\mathsf{LANG}_k(A_{k+1}) \subseteq \pi_{A_{k+1}}(\mathsf{REC}_{k+1})$.*

Preuve : Soit $\mathcal{A} \in k\text{-}\mathsf{AP}^{\vec{\emptyset}}(A_{k+1}(A_1, \ldots, A_k))$, nous construisons un nouvel automate, $\mathcal{B} \in (k+1)\text{-}\mathsf{AP}^{\vec{\emptyset}}(A_{k+1}, (A_1, \ldots, A_k, A_{k+1}))$, reconnaissant le même langage que $\mathcal{A}$, et vérifiant : $\forall q \in Q$, $\sigma \in {A_{k+1}}^*$, $\omega \in k+1\text{-}\mathsf{Pile}$:

$$(q_0, \sigma, \bot_{k+1}) \vdash_{\mathcal{B}}^* (q, \varepsilon, \omega) \text{ implique } \mathrm{mot}_{k+1}(\omega) = \sigma.$$

Notons que dans ce cas, d'après la remarque 1.4.11, $\sigma = \pi_{A_{k+1}}(\varphi_{k+1}(\omega))$.

Pour obtenir un tel automate, il suffit de substituer toute transition $\delta = (p, \alpha, w, \mathrm{instr}, q)$ de $\mathcal{A}$ par l'ensemble de transitions

$$\delta' = \{(p, \alpha, \beta w, \mathrm{instr}\, \alpha\, \mathrm{instr}, q) \mid \beta \in A_{k+1}\},$$

où $\mathrm{instr}_\alpha = \mathrm{push}_{k+1,\alpha}$ si $\alpha \in A_{k+1}$ et $\mathrm{instr}_\varepsilon = \mathrm{stay}$.

L'ensemble de pile $P = \mathrm{ACC}(\mathcal{B}) \in \mathsf{PREC}_{k+1}$ généré par cet automate vérifie : $\mathrm{L}(\mathcal{A}) = \mathrm{mot}_{k+1}(P) = \pi_{A_{k+1}}(\varphi_k(P))$. D'après la proposition 10.1.2, $P \in \mathsf{PREC}_{k+1}$ implique $\varphi_k(P) \in \mathsf{REC}_{k+1}$ d'où

$$\mathrm{L}(\mathcal{A}) \in \pi_{A_{k+1}}(\mathsf{REC}_{k+1}).$$

□

10.1.2 Problèmes de reconnaissabilité

Il est ainsi possible de contrôler les transitions d'un k-AP par des ensembles définissables en LSOM dans la structure associée à son graphe de calcul.

Ceci permet par exemple, de construire pour tout k-AP un nouveau k-AP équivalent *émondé*, c'est à dire dans lequel tout calcul "commencé" aboutit à un calcul acceptant.

Définition 10.1.6 (Automate émondé). *Un k-AP $\mathcal{A} = (Q, \Sigma, (A_1, \ldots, A_k), \vec{C}, \Delta, q_0, \bot, F)$ est dit **émondé**, si pour tous $\sigma \in \Sigma^*$, $p \in Q$ et $\omega \in k\text{-}\mathsf{Pile}$,*

$$(q_0, \sigma, \bot_k) \vdash_+^{\mathcal{A}} (p, \varepsilon, \omega) \textit{ implique } \exists \sigma' \in \Sigma^*,\ \exists q \in F, \omega' \in k\text{-}\mathsf{Pile} \textit{ tels que } (p, \sigma', \omega) \vdash_{\mathcal{A}}^* (q, \varepsilon, \omega')$$

(On a alors $\sigma\sigma' \in \mathrm{L}(\mathcal{A})$).

Proposition 10.1.7. *Tout langage dans LANG_k est reconnu par automate dans $k\text{-}\mathsf{AP}^{\vec{\emptyset}}$ émondé.*

Preuve : Soit $\mathcal{A} = (Q, \Sigma, (A_1, \ldots, A_k), \Delta, q_0, \bot)$ un k-AP non contrôlé. Définissons pour tout $p \in Q$, l'ensemble P_p des k-piles ω telles qu'il existe un calcul issu de (p, ω) et aboutissant à une configuration totale acceptante :

$$P_p = \{\omega \in k\text{-}\mathsf{Pile} \mid \exists q \in F,\ \omega' \in k\text{-}\mathsf{Pile},\ (p, \omega) \vdash_{\mathcal{A}}^* (q, \omega')\}.$$

Chaque P_p est clairement définissable en LSOM dans $\mathbf{Pile_k}$, et appartient donc à PREC_k. Supposons que $Q = \{p_1, \ldots, p_m\}$ et définissons le k-AP contrôlé

$$\mathcal{A}^{\vec{P}} = (Q, \Sigma, (A_1, \ldots, A_k), (P_{p_1}, \ldots, P_{p_m}), \Delta', q_0, \bot)$$

où Δ' est composé de toutes les transitions $(p_i, \alpha, w, \vec{o}, \text{instr}, q)$ telles que

$$\pi_i(\vec{o}) = 1 \text{ et } (p_i, \alpha, w, \text{instr}, q) \in \Delta.$$

Clairement, l'automate $\mathcal{A}^{\vec{P}} \in k\text{-}\mathsf{AP}^{\mathsf{PREC}_k}$ est émondé et reconnaît le même langage que $\mathcal{A}$. D'après le théorème 9.3.7, il existe $\mathcal{B} \in k\text{-}\mathsf{AP}^{\emptyset}$, tel que $\mathcal{A}^{\vec{P}} \leq \mathcal{B}$, et par définition de simulation déterministe, l'automate $\mathcal{B}$ est également émondé et reconnaît le langage $\mathcal{A}$.

□

Un corollaire immédiat de cette proposition est que pour tout langage L reconnu par un k-AP, l'ensemble des préfixes de L est également reconnu par un k-AP.

Corollaire 10.1.8. *Pour tout L appartenant à $\mathsf{LANG}_k(\Sigma)$, $k \geq 1$, l'ensemble des préfixes de L $FG(L) = \{u \in \Sigma^* \mid \exists v \in \Sigma^*, uv \in L\}$ appartient à $\mathsf{LANG}_k(\Sigma)$.*

Preuve : Soit $L \in \mathsf{LANG}_k(\Sigma)$. D'après la proposition précédente, nous pouvons supposer que L est reconnu par un automate $\mathcal{A} \in k\text{-}\mathsf{AP}^{\vec{\emptyset}}$ émondé. Alors, l'automate obtenu en déclarant dans $\mathcal{A}$ tous les états comme finaux, reconnaît le langage $FG(L)$.

□

Le même type de construction que celui donné dans le corollaire permet de montrer deux autres résultats :

- le *centre* (voir [BN81]) d'un langage L dans DLANG_k est également dans LANG_k,
- pour tout $L = \{u_1, \ldots, u_n, \ldots\} \in \mathsf{LANG}_k$ où chaque u_i est préfixe de u_{i+1}, L est reconnu par un k-AP déterministe.

Définition 10.1.9 (Centre d'un langage). *Soit L un langage sur un alphabet Σ. Nous appelons centre de L le langage*

$C(L) = \{\sigma \in \Sigma^* \mid \exists$ *une infinité de* $\sigma' \in \Sigma^*$ *tels que* $\sigma\sigma' \in L\}$

Proposition 10.1.10. *Pour tout $L \in \Sigma^*$ et $k \geq 1$:*

1. *Si $L \in \mathsf{DLANG}_k$, le centre de L, $C(L)$ appartient à DLANG_k.*
2. *Si $L \in \mathsf{LANG}_k$ et il existe $\sigma_1, \ldots, \sigma_n, \ldots \in \Sigma^*$ tels que*

$$L = \{\varepsilon, \sigma_1, \sigma_1\sigma_2, \ldots, \sigma_1\sigma_2 \ldots \sigma_n, \ldots\},$$

alors il existe un k-AP déterministe non contrôlé reconnaissant L.

Preuve : Soit $\mathcal{A} = (Q, \Sigma, (A_1, \ldots, A_k), \Delta, q_0, \bot, F) \in k\text{-}\mathsf{AP}^{\vec{\emptyset}}$ reconnaissant un langage L. Commençons par montrer que, pour tout $p \in Q$, l'ensemble $R_p = \{\omega \in k\text{-}\mathsf{Pile} \mid$ il existe $\sigma \in \Sigma^\omega$ et un calcul de σ issu de (p, ω) et passant une infinité de fois par des configurations acceptantes$\}$ est définissable en LSOM dans $\mathbf{Pile_k}$.

Supposons que $Q = \{p_1, \ldots, p_m\}$, et définissons la formule dans SOM($\mathbf{Pile_k}$) suivante :

$$\mathbf{P}(X_{p_1}, \ldots, X_{p_m}) := \bigwedge_{q \in Q} \forall x, [x \in X_q \Leftrightarrow \bigvee_{p \in F} [\exists y \in X_p,\ (q, x) \overset{\sigma \neq \varepsilon}{\rightarrow}_{\mathcal{A}} (p, y)]].$$

Clairement, $\mathbf{Pile_k}, (R_{p_1}, \ldots, R_{p_m}) \models \mathbf{P}(X_{p_1}, \ldots, X_{p_m})$.

Remarquons que cette formule admet une plus grande solution. En effet, pour tous vecteurs $(R_{p_1}, \ldots, R_{p_m})$ et $(R'_{p_1}, \ldots, R'_{p_m})$, satisfaisant $\mathbf{P}$ dans $\mathbf{Pile_k}$, le vecteur $(R_{p_1} \cup R'_{p_1}, \ldots, R_{p_m} \cup$

R'_{p_m}) est trivialement une solution de $\mathbf{P}$ dans $\mathbf{Pile_k}$. Il existe donc une plus grande solution consistant en l'union de toutes les solutions. Enfin, on voit clairement que pour toute solution de $\mathbf{P}$, il existe un calcul infini lisant un mot infini et passant une infinité de fois par une configuration acceptante, et donc pour tous ensembles de piles $S_{p_1}, \ldots, S_{p_m}$, $\mathbf{Pile_k}, (S_{p_1}, \ldots, S_{p_m}) \models \mathbf{P}(X_{p_1}, \ldots, X_{p_m})$ implique $S_{p_1} \subseteq R_{p_1}, \ldots, S_{p_m} \subseteq R_{p_m}$.

Il existe donc une plus grande solution, et cette solution est le vecteur $(R_{p_1}, \ldots, R_{p_m})$. Posons

$$\cup\mathbf{P}(X_{p_1}, \ldots, X_{p_m}) := \forall Y_1, \ldots, Y_m, [\mathbf{P}(Y_1, \ldots, Y_m) \implies \bigwedge_{i \in [1,m]} Y_i \subseteq X_{p_i}.$$

D'après ce qui précède, $\cup\mathbf{P}$ admet une unique solution dans $\mathbf{Pile_k}$, correspondant au vecteur $\vec{R} = (R_{p_1}, \ldots, R_{p_m})$.

Donc, pour tout $p \in P$, l'ensemble R_p appartient à PREC_k. Soit $\mathcal{A}^{\vec{R}}$ l'automate obtenu en appliquant à $\mathcal{A}$ la construction donnée dans la preuve du corollaire 10.1.7 mais ayant pour contrôleur le vecteur $\vec{R}$.

1. Si $\mathcal{A}$ est déterministe, alors $\mathcal{A}^{\vec{R}}$ est déterministe et pour tous $\sigma \in \Sigma^*$, $p \in Q$ et $\omega \in k\text{-}\mathsf{Pile}$, si, $(q_0, \sigma, \bot_k) \rightarrow^+_{\mathcal{A}^{\vec{R}}} (p, \varepsilon, \omega)$, alors il existe une infinité de $\sigma' \in \Sigma^*$ pour lesquels il existe $q \in Q$ tel que (p, σ', ω) aboutit à une configuration acceptante.

   L'automate $\mathcal{B}$ obtenu en déclarant tous les états de $\mathcal{A}^{\vec{R}}$ comme finaux reconnaît le langage $C(L)$. En appliquant le théorème 9.3.7 à $\mathcal{B}$, nous obtenons un k-AP sans oracle reconnaissant $C(L)$.

2. Si $L = \{\varepsilon, \sigma_1, \sigma_1\sigma_2, \ldots, \sigma_1\sigma_2 \ldots \sigma_n, \ldots\}$, alors pour toute paire de transitions compatibles (voir définition 2.4.1), $(p, \alpha_1, w, f, \vec{o}, q)$, $(p, \alpha_2, w, f', \vec{o}, q')$ de $\mathcal{A}^{\vec{R}}$, l'application de chacune de ces transitions à une configuration accessible (p, ε, ω) aboutit à un calcul du même mot infini.

   Donc si on supprime une des deux transitions, l'automate reconnaît encore le même langage. Nous obtenons donc un automate déterministe en appliquant la procédure suivante :

   Tant que Δ est non déterministe :
   – choisir une paire de transitions compatible dans Δ,
   – en supprimer une des deux.
   Fin.

   Nous obtenons ainsi un automate $\mathcal{A}_1$ déterministe, contrôlé par $\vec{R}$ et reconnaissant le langage L. D'après le théorème 9.3.7, et les propriétés de la simulation déterministe, L est reconnu par un k-AP non contrôlé déterministe.

□

10.1.3 Problèmes de décision

On s'intéresse maintenant à la complexité en temps de la relation d'accessibilité d'un k-AP.

Soit $\mathcal{A}$ un k-AP et étant données deux configurations totales (p, ω) et (q, ω') de $\mathcal{A}$. Nous savons déjà décider si $(p, \omega) \vdash^*_{\mathcal{A}} (q, \omega')$ puisque cette propriété est exprimable dans $\mathbf{Pile_k}$ qui admet une SOM-théorie décidable (voir théorème 3.4.1). Le codage des éléments de k-Pile par des éléments de $\mathcal{P}_k$ permet de résoudre le problème en temps linéaire.

Proposition 10.1.11. *Soit $\mathcal{A}$ un k-AP fixé et (p,ω) et (q,ω') deux états totaux quelconques de $\mathcal{A}$.*

Alors le problème $(p,\omega) \overset{*}{\rightarrow}_{\mathcal{A}} (q,\omega')$ est décidable en temps linéaire par rapport à $|\varphi_k(\omega)|$, $|\varphi_k(\omega')|$.

Preuve : Puisque le relation d'accessibilité de $\mathcal{A}$ est SOM-définissable dans $\mathbf{Pile_k}$ alors en utilisant le théorème 6.2.1), son image par φ_k est SOM-définissable dans $\boldsymbol{\mathcal{P}_k}$. Formellement, il existe une formule $Acc_{p,q}(x,y)$ vérifiant pour tout $u,v \in \mathcal{P}_k$,

$$\boldsymbol{\mathcal{P}_k} \models Acc_{p,q}(u,v) \text{ ssi } (p,\varphi_k^{-1}(u)) \vdash_{\mathcal{A}}^{*} (q,\varphi_k^{-1}(v))$$

D'après la proposition 5.1.3, il existe une formule ${Acc_{p,q}}^{+1}$ vérifiant pour tout $u,v \in \mathcal{P}_{k+1}$,

$\boldsymbol{\mathcal{P}_{k+1}} \models {Acc_{p,q}}^{+1}(u,v)$ ssi $Acc_{p,q}(\mathrm{f}_k(u),\mathrm{f}_k(v))$

Soit $\# \in A_{k+1}$, d'après ce qui précède, langage $S = \{u\#\overline{u} \bullet v |\ u,v \in \mathcal{P}_k \text{ et } Acc_{p,q}(u,v)\}$ appartient à REC_{k+1}, puisqu'il correspond à l'ensemble des $v \in \mathcal{P}_{k+1}$ tels que

$\exists u \in \mathcal{P}_k$ tel que $w \in u\#\mathrm{Irr}(A_k)$ et $\boldsymbol{\mathcal{P}_k} \models Acc_{p,q}(f_k(u),f_k(w))$.

Alors pour tout $\omega,\omega' \in k\text{-}\mathsf{Pile}$ $(p,\omega) \vdash_{\mathcal{A}}^{*} (q,\omega')$ ssi $\varphi_k^{-1}(\omega)\#\overline{\varphi_k^{-1}(\omega)} \bullet \varphi_k^{-1}(\omega') \in S$. Comme $S \in \mathsf{REC}_{k+1}$, d'après la proposition 10.1.2, ce problème est décidable en temps nk, où n est la taille du mot à tester.

□

Proposition 10.1.12 (Problèmes sur les langages dans LANG_k). *Soit $k \geq 1$ et $\mathcal{A} \in k\text{-}\mathsf{AP}^{\vec{\emptyset}}$ sur l'alphabet d'entrée Σ et $L = \mathrm{L}(\mathcal{A})$.*

1. *Étant donné σ un mot dans Σ^*, le problème "$u \in L$?" est décidable.*
2. *Le problème "$L = \emptyset$?" est décidable.*
3. *Si $\mathcal{A}$ est* déterministe, *le problème "L est-il fini ?" est décidable.*
4. *Si $\mathcal{A}$ est* déterministe, *alors étant donné σ un mot dans Σ^*, on sait décider si le résiduel $u^{-1}L = \{uv \in L\}$ est fini.*

Preuve :

1. Pour tout $\sigma \in \Sigma$, il est possible de construire une formule dans le graphe de calcul de $\mathcal{A}$ vrai ssi $\sigma \in L$.
2. $L = \emptyset$ ssi l'ensemble des préfixes de L ne contient pas ε. D'après la proposition 10.1.8, l'ensemble $FG(L)$ est reconnu par automate dans $k\text{-}\mathsf{AP}^{\vec{\emptyset}}$ et en utilisant le point précédent, nous savons donc décider si $L = \emptyset$.
3. L est fini ssi le centre de L est vide (voir définition 10.1.9). D'après la proposition 10.1.10, $C(L)$ est reconnu par un automate dans $k\text{-}\mathsf{AP}^{\vec{\emptyset}}$ et en utilisant le point précédent, nous savons donc décider si L est fini.
4. $u^{-1}L$ est fini ssi $u \notin C(L)$, en utilisant encore la proposition 10.1.10 et le point 1 de la proposition, nous savons donc décider si $u^{-1}L$ est fini.

□

Remarque 10.1.13. *Le point (2) de la proposition 10.1.12 est énoncé dans [Gre70, page 12, lignes 32-33, attribué à Aho-Ullman], dans [Mas74, page 1171, ligne 28] et entièrement prouvé dans [Dam82, théorèmes 7.8 and 7.17].*

La complexité précise du problème 2 est déterminée dans [Eng91, théorème 7.12 p.71] : le problème du vide pour un k-AP est $DTIME(exp_{k-1}(\mathcal{O}(n^2)))$-complet.

Le point (1) découle aisément du point (2) et de la clôture effective des langages de niveau k par intersection avec des langages réguliers.

Le point (3) est établi, mais seulement pour les niveaux inférieurs à 2 dans [Aya73, corollaire 5.1], par l'application d'un lemme d'itération.

10.2 Cônes rationnels de niveau k

Cette partie utilise la notion de transduction rationnelle qu'on rappelle brièvement ici. Le lecteur pourra se référer par exemple à [AB88, Ber79] pour une introduction complète.

10.2.1 Transductions rationnelles.

Une *transduction* τ de X^* dans Y^* est une application de X^* dans l'ensemble $\mathrm{P}(Y^*)$ des sous-ensembles de Y^*. Par commodité, on écrit $\tau : X^* \to Y^*$. Ainsi, τ peut être vue comme un sous-ensemble de $X^* \times Y^*$.

Définition 10.2.1. *Soit M un monoïde. La famille $Rat(M)$ des sous-ensembles rationnels de M est la plus petite famille $\mathcal{R}$ de sous-ensembles de M satisfaisant les conditions suivantes :*
- *$\emptyset \in \mathcal{R}$, $\{m\} \in \mathcal{R}$ pour tout $m \in M$,*
- *si $A, B \in \mathcal{R}$, alors $A \cup B$, $AB \in \mathcal{R}$,*
- *si $A \in \mathcal{R}$, alors $A^+ = \bigcup_{n \geq 1} A^n \in \mathcal{R}$.*

Définition 10.2.2 (Transduction rationnelle). *Une transduction $\tau \subseteq X^* \times Y^*$ est une **transduction rationnelle** si c'est un sous-ensemble rationnel de $X^* \times Y^*$.*

Si on choisit une représentation fonctionnelle de $\tau : X^* \to Y^*$, alors τ est rationnelle si son graphe de relations $R = \{(f, g) \in X^* \times Y^* | \, g \in \tau(f)\}$ est un sous-ensemble rationnel $X^* \times Y^*$.

Le théorème suivant est dû à Nivat [Niv68] et est également prouvé dans [AB88, p.23, théorème 3.1.c].

Théorème 10.2.3. *Soit X et Y deux alphabets. Les conditions suivantes sont équivalentes*

1. *$\tau : X^* \to Y^*$ est rationnelle,*
2. *il existe un alphabet Z, deux morphismes $\varphi : Z^* \to X^*$, $\psi : Z^* \to Y^*$ et un langage régulier $K \subseteq Z^*$ tel que*

$$\forall u \in X^* \; \tau(u) = \psi(\varphi^{-1}(u) \cap K).$$

Définition 10.2.4 (Domination rationnelle). *Soient L et L' deux langages. On dit que L **domine rationnellement** L' si il existe une transduction rationnelle τ telle que $L' = \tau(L)$. On écrit alors $L' \leq L$.*

Définition 10.2.5 (Cône rationnel). *Un **cône rationnel** est une famille de langages fermée par transduction rationnelle.*

En d'autres mots, une famille $\mathcal{L}$ de langages est un cône si il satisfait la condition suivante :

$$\textit{si } L \in \mathcal{L} \textit{ et } L' \leq L \textit{ alors } L' \in \mathcal{L}.$$

Un corollaire immédiat du théorème 10.2.3 est le suivant :

Lemme 10.2.6. *Une famille de langages est un cône rationnel ssi elle est close par homomorphisme, homomorphisme inverse et intersection avec un ensemble régulier.*

Ces propriétés ont déjà été étudiées pour les langages de niveau k, Maslov prouve que les langages de niveau k vérifient ces trois propriétés de clôture [Mas74, p. 1172].

Théorème 10.2.7. *Pour tout $k \geq 1$, la famille LANG_k est un cône rationnel. En d'autres mots, pour tout $L \in \mathsf{LANG}_k$, pour toute transduction rationnelle τ, $\tau(L) \in \mathsf{LANG}_k$.*

Définition 10.2.8 (Cône principal). *Un cône rationnel $\mathcal{L}$ est dit **principal** si il existe $L_0 \in \mathcal{L}$ appelé **générateur** de $\mathcal{L}$ tel que*

$$\text{pour tout } L \in \mathcal{L},\ L \leq L_0.$$

10.2.2 Générateur de LANG_k

Nous prouvons ici que la classe LANG_k est générée par le langage $\mathcal{M}_k$. Nous prouvons pour cela, que pour tout langage L de niveau k, il existe une transduction rationnelle τ telle que $\tau(\mathcal{M}_k) = L$.

Théorème 10.2.9. *Pour tout $L \in \mathsf{LANG}_k(\Sigma)$, $k \geq 1$, il existe une transduction rationnelle τ telle que $\tau(\mathcal{M}_k) = L$.*

Construction : D'après le théorème 8.2.4, il existe un automate $\mathcal{A} \in k\text{-}\mathsf{N1AP}(A_1, \ldots, A_k)$, non contrôlé et reconnaissant L. Posons $B_1 = Q \cup \{\bot\} \cup A_1$ et $B_i = A_i$ pour $i \geq 2$. Nous définissons une transduction rationnelle $\tau \subseteq (\bigcup_{i\in[1,k]} B_i \cup \overline{B_i})^* \times \Sigma^*$ vérifiant pour tout $\omega \in k\text{-}\mathsf{Pile}(A_1, \ldots, A_k)$: $(q_0, \sigma, \bot_k) \vdash_{\mathcal{A}}^* (p, \varepsilon, \omega)$ ssi il existe $v \in \mathcal{M}_k(B_1, \ldots, B_k)$ tel $\tau_\delta(w) = \sigma$. Considérons donc les transductions suivantes :

- $\tau_{\mathrm{push}_{i,a}} = (a, \varepsilon)$,
- $\tau_{\mathrm{pop}_i} = \{(b, \varepsilon) \mid b \in \widehat{A_{i-1}}\}^* \cdot \{(\bar{b_i}, \alpha) \mid b_i \in A_i\}$.

Pour toute transition $\bar{\delta}$ représentée par $\delta = (p, \alpha, a_1, \mathrm{instr}, q)$, nous posons

$$\tau_\delta = (\bar{p}\bar{a_1}a_1, \varepsilon) \cdot \tau_{\mathrm{instr}} \cdot (q, \alpha).$$

Définissons finalement les transductions

$$\tau_q = (\bot\ q_0, \varepsilon) \cdot (\bigcup_{\bar{\delta}\in\Delta} \tau_\delta)^*(\overline{q}, \varepsilon) \text{ et } \tau = \bigcup_{q\in F} \tau_q.$$

Remarque 10.2.10. *Pour prouver la validité de la construction, nous allons utiliser à nouveau le codage φ_k des éléments de k-Pile sous forme de mots dans $\mathcal{P}_k$. Rappelons ici les résultats sur $\mathcal{P}_k$, $\mathcal{M}_k$ et φ_k qui seront utilisées au cours de la preuve :*

1. *par définition, pour tout $u \in \mathcal{M}_k(A_1, \ldots, A_k)$, $\rho(u) \in \mathcal{P}_k(A_1, \ldots, A_k)$, et pour tout $v \preccurlyeq u$, $v \in \mathcal{M}_k(A_1, \ldots, A_k)$,*
2. *pour tout $u \in \mathcal{M}_k(A_1, \ldots, A_k)$, $a \in A_i$,*
   $ua \in \mathcal{M}_k$ et $[u\bar{a} \in \mathcal{M}_k$ ssi $\mathrm{fin}_i\, \rho(u) = a]$,
3. *pour tout $\omega \in k$-Pile, tel que $\mathrm{top}_1(\omega) = a \in A_1 \cup \{\bot\}$,*
   si $a \in A_1$, alors $a = \mathrm{fin}_1(\varphi_k(\omega))$, sinon $\mathrm{fin}_1(\varphi_k(\omega)) = \varepsilon$,

4. pour tout $a \in A_i$, $\omega, \omega' \in k\text{-Pile}(A_1, \ldots, A_k)$,
$\omega' = \mathrm{push}_a(\omega)$ *ssi* ${\varphi_k}^{-1}(\omega') = {\varphi_k}^{-1}(\omega) \bullet a$, *et*
$\omega' = \mathrm{pop}_i(\omega)$ *ssi* $\exists w \in \widehat{A_{i-1}}^*$, $b \in A_i$, ${\varphi_k}^{-1}(\omega') = {\varphi_k}^{-1}(\omega) \bullet w \bullet \bar{b}$.

La validité de cette construction est prouvée par le lemme 10.2.15, qui découle des lemmes intermédiaires 10.2.11, 10.2.12 et 10.2.14.

Lemme 10.2.11. *Pour tous* $u_1, u_2 \in \mathcal{P}_k(A_1, \ldots, A_k)$, *pour tout* $v \in \mathcal{M}_k$ *tel que* $\rho(v) =\!\!\perp u_1$,

$$ {\varphi_k}^{-1}(u_2) = \mathrm{instr}({\varphi_k}^{-1}(u_1)) \text{ ssi } \exists w,\ \tau_\delta(w) = \varepsilon,\ vw \in \mathcal{M}_k \text{ et } \rho(vw) =\!\!\perp u_2. $$

Preuve : Distinguons deux cas selon la valeur de instr :
- si $\mathrm{instr} = \mathrm{push}_{a,i}$, d'après la remarque 10.2.10(4),

$$ \begin{array}{lll} {\varphi_k}^{-1}(u_2) = \mathrm{instr}({\varphi_k}^{-1}(u_1)) & \text{ssi} & u_2 = u_1 \bullet a \\ & \text{ssi} & va \in \mathcal{M}_k,\ \tau_{\mathrm{push}_{i,a}}(a) = \varepsilon \text{ et } \rho(va) =\!\!\perp u_1 \bullet a =\!\!\perp u_2 \end{array} $$

- si $\mathrm{instr} = \mathrm{pop}_i$, alors d'après la remarque 10.2.10(4), ${\varphi_k}^{-1}(u_2) = \mathrm{pop}_i({\varphi_k}^{-1}(u_1))$ ssi il existe $w \in \mathrm{Irr}_{i-1} \cdot \overline{A_i}$ tel que $u_2 = u_1 \bullet w$. Par définition de τ_{pop_i}, pour tout w, $\tau_{\mathrm{pop}_i}(w) = \varepsilon$ ssi $w \in \widehat{A_{i-1}}^* \overline{A_i}$. Nous obtenons donc,

$$ \begin{array}{lll} {\varphi_k}^{-1}(u_2) = \mathrm{pop}_i({\varphi_k}^{-1}(u_1)) & \text{ssi} & \exists w,\ vw \in \mathcal{M}_k,\ u_1 \bullet \rho(w) = u_2 \text{ et } \tau_\delta(w) = \varepsilon \\ & \text{ssi} & w,\ \tau_\delta(w) = \varepsilon,\ vw \in \mathcal{M}_k \text{ et } \rho(vw) =\!\!\perp u_2. \end{array} $$

□

Lemme 10.2.12. *Pour tous* $u_1, u_2 \in \mathcal{P}_k(A_1, \ldots, A_k)$, *pour tout* $v \in \mathcal{M}_k$ *tel que* $\rho(v\bar{p}) =\!\!\perp u_1$,

$$ (p, \alpha, {\varphi_k}^{-1}(u_1)) \vdash_\delta (q, \varepsilon, {\varphi_k}^{-1}(u_2)) \text{ ssi } \exists w,\ \tau_\delta(w) = \alpha,\ vw \in \mathcal{M}_k \text{ et } \rho(vw) =\!\!\perp u_2 $$

Preuve : Supposons $\delta = (p, \alpha, a, \mathrm{instr}, q)$, $a \in A_1 \cup \{\perp\}$,

$$ (p, \alpha, {\varphi_k}^{-1}(u_1)) \vdash_\delta (q, \varepsilon, {\varphi_k}^{-1}(u_2)) \text{ ssi } \mathrm{top}_1({\varphi_k}^{-1}(u_1)) = a \text{ et } {\varphi_k}^{-1}(u_2) = \mathrm{instr}({\varphi_k}^{-1}(u_1)). \quad (14) $$

Or,

$$ \mathrm{top}_1({\varphi_k}^{-1}(u_1)) = a \text{ ssi } \mathrm{fin}_1(\rho(v\bar{p})) = a \text{ ssi } v\bar{p}\bar{a} \in \mathcal{M}_k \text{ ssi } v' = v\bar{p}\bar{a}a \in \mathcal{M}_k. $$

Alors, $\rho(v') =\!\!\perp u_1$ et d'après le lemme 10.2.11,

$$ {\varphi_k}^{-1}(u_2) = \mathrm{instr}({\varphi_k}^{-1}(u_1)) \text{ ssi } \exists w,\ \tau_{\mathrm{instr}}(w) = \varepsilon,\ v'w \in \mathcal{M}_k \text{ et } \rho(v'w) =\!\!\perp u_2. $$

En substituant ces deux équivalences dans le second membre de (14), nous obtenons

$$ (p, \alpha, {\varphi_k}^{-1}(u_1)) \vdash_\delta (q, \varepsilon, {\varphi_k}^{-1}(u_2)) \text{ ssi } \exists w,\ \tau_\delta(w) = \alpha,\ v'w \in \mathcal{M}_k \text{ et } \rho(v'w) =\!\!\perp u_2. $$

□

Avant d'étudier le langage définit par une transduction de type τ_p, il est utile de remarquer la propriété suivante :

Remarque 10.2.13. *Pour tout* $w \in \mathcal{M}_k$, $\sigma \in \Sigma^*$, $q \in Q$,
$\tau_q(w) = \sigma$ *ssi*
- *soit* $w = \perp q_0\bar{q_0}$ *et* $\sigma = \varepsilon$,
- *soit il existe* w_1, w_2 *et* $\delta = (p, \alpha, a, \text{instr}, q)$, $\sigma' \in \Sigma^*$ *tels que* $w = w_1 w_2 \bar{q}$, $\tau_p(w_1\bar{p}) = \sigma'$, $\tau_\delta(w_2) = \alpha$ *et* $\sigma = \sigma'\alpha$.

Lemme 10.2.14. *Pour tout* $u \in \mathcal{P}_k(A_1, \ldots, A_k)$, $\sigma \in \Sigma^*$, $q \in Q$,

$$(q_0, \sigma, \perp_k) \vdash_{\mathcal{A}}^* (q, \varepsilon, {\varphi_k}^{-1}(u)) \text{ ssi } \exists v \in \mathcal{M}_k,\ \rho(v) = \perp u \text{ et } \tau_q(v) = \sigma.$$

Preuve : Prouvons le lemme par induction sur $n \geq 0$, où n est la longueur du calcul dans $\mathcal{A}$.
Base : si $n = 0$, alors

$$(q_0, \sigma, \perp_k) \vdash_{\mathcal{A}}^0 (q, \varepsilon, {\varphi_k}^{-1}(u)) \text{ ssi } q = q_0 \text{ et } u = \varepsilon \text{ ssi } \tau_q(\perp q_0\bar{q_0}) = \sigma.$$

Pas d'induction : supposons le lemme vrai pour $n \geq 0$. Soient $u_1, u_2 \in \mathcal{P}_k(A_1, \ldots, A_k)$, $\sigma, \alpha \in \Sigma^*$, et $\delta = (p, \alpha, a, \text{instr}, q)$. Supposons

$$(q_0, \sigma\alpha, \perp_k) \vdash_n^{\mathcal{A}} (p, \alpha, {\varphi_k}^{-1}(u_1)) \vdash_\delta (q, \varepsilon, {\varphi_k}^{-1}(u_2)). \tag{15}$$

En appliquant l'hypothèse d'induction à u_1, nous obtenons

$$(15) \quad \text{ssi} \quad \exists v_1' \in \mathcal{M}_k,\ \rho(v_1') = \perp u_1 \text{ et } \tau_p(v_1') = \sigma.$$

Or, $\tau_p(v_1') = \sigma$ implique $v_1' = v_1\bar{p}$. Donc

$$(15) \quad \text{ssi} \quad \exists v_1 \in \mathcal{M}_k,\ \rho(v_1\bar{p}) = \perp u_1,\ \tau_p(v_1\bar{p}) = \sigma \text{ et } (p, \alpha, {\varphi_k}^{-1}(u_1)) \vdash_\delta (q, \varepsilon, {\varphi_k}^{-1}(u_2)).$$

En appliquant le lemme 10.2.12,

$$(15) \quad \text{ssi} \quad \exists v_2 = v_1 w \in \mathcal{M}_k,\ \tau_p(v_1\bar{p}) = \sigma,\ \tau_\delta(w) = \alpha \text{ et } \rho(v_2) = \perp u_2.$$

Finalement d'après la remarque précédent le lemme, nous obtenons

$$(15) \quad \text{ssi} \quad \exists v_2 \in \mathcal{M}_k,\ \rho(v_2) = \perp u_2 \text{ et } \tau_q(v_2) = \sigma\alpha.$$

□

Achevons maintenant la preuve de la proposition :

Lemme 10.2.15. *Pour tout* $\sigma \in \Sigma^*$,

$$\exists w \in \mathcal{M}_k,\ \tau(w) = \sigma \text{ ssi } \sigma \in \mathrm{L}(\mathcal{A}).$$

Preuve : Par définition de τ, $\tau(w) = \sigma$ ssi il existe $q \in F$ tel que $\tau_q(w) = \sigma$. Donc, en utilisant le lemme 10.2.14, nous obtenons

$$\begin{aligned} \sigma \in \mathrm{L}(\mathcal{A}) \quad & \text{ssi} \quad \exists q \in F, u \in \mathcal{P}_k,\ (q_0, \sigma, \perp_k) \vdash_{\mathcal{A}}^* (q, \varepsilon, {\varphi_k}^{-1}(u)) \\ & \text{ssi} \quad \exists v \in \mathcal{M}_k,\ \exists q \in F, \exists u \in \mathcal{P}_k(A_1, \ldots, A_k),\ \rho(v) = \perp u \text{ et } \tau_q(v) = \sigma \\ & \text{ssi} \quad \exists w \in \mathcal{M}_k,\ \tau(w) = \sigma. \end{aligned}$$

□

Nous venons donc de prouver que pour tout $L \in \mathsf{LANG}_k$, *il existe des alphabets* $B_1, \ldots, B_k$ et une transduction rationnelle τ telle que $\tau(\mathcal{M}_k(B_1, \ldots, B_k)) = L$. Pour montrer que LANG_k est un cône principal, il faut pouvoir fixer une famille $A_1, \ldots, A_k$ de telle façon que pour tout $L \in \mathsf{LANG}_k$, il existe une transduction rationnelle τ telle que $\tau(\mathcal{M}_k(A_1, \ldots, A_k)) = L$.

Dans la preuve du théorème, nous avons supposé les alphabets $A_2, \ldots, A_k$ quelconques, mais puisque la construction est effectuée à partir d'un automate N1-normalisé, rien n'empêche de considérer que les alphabets $B_i = A_i$, pour $i \geq 1$ sont réduits à l'unique élément $\#_i$. Prouvons maintenant qu'il est possible de trouver un alphabet B_1^{min} minimum tel que, pour tout langage $L \in \mathsf{LANG}_k$, il existe une transduction rationnelle sur les alphabets $B_1^{min}, \{\#_2\}, \ldots \{\#_k\}$ reconnaissant ce langage.

Définition 10.2.16. *Pour tout $n \geq 2$, pour tout A_1 tel que $|A_1| = n$, on note $\mathcal{M}_{k,n} = \mathcal{M}_k(A_1, \{\#_2\}, \ldots, \{\#_k\}))$.*

Corollaire 10.2.17. *Pour tout $L \in \mathsf{LANG}_k$, il existe $n \geq 2$ et une transduction rationnelle τ telle que $\tau(\mathcal{M}'_{k,n}) = L$.*

Lemme 10.2.18. *Soit $n \geq 2$, $k \geq 1$, il existe une transduction rationnelle τ telle que $\mathcal{M}_{k,n} = \tau(\mathcal{M}_{k,2})$.*

Preuve : Supposons $\mathcal{M}_{k,2}$ est écrit avec $B_1 = \{a, b\}$ et que $\mathcal{M}_{k,n}$ est écrit sur $A_1 = \{a_1, \ldots, a_n\}$. Nous définissons un homomorphisme $\psi : \widehat{\Gamma}_1^* \mapsto \widehat{\Gamma}_2^*$ par $\psi(a_i) = ab^i a$ et $\psi(\overline{a_i}) = \overline{a}\overline{b}^i\overline{a}$ pour tout $a_i \in A_1$ et $\psi(x) = x$ sinon.

Une simple vérification montre que $\mathcal{M}_{k,n} = \psi^{-1}(\mathcal{M}_{k,2})$, donc par le théorème de Nivat (théorème 10.2.3), en posant $\tau = \psi^{-1}$, le lemme est vérifié.

□

Il est énoncé dans [Mas76, théorème 4] que pour tout $k \geq 1$, les langages de niveau k forment un cône principal. Nous en donnons ici un générateur.

Théorème 10.2.19. *Pour tout $k \geq 1$, LANG_k est un cône principal généré par le langage $\mathcal{M}_{k,2}$.*

Preuve : Pour tout $k \geq 1$, $n \geq 2$, d'après le lemme 9.1.10, la langage $\mathcal{M}_{k,n}$ est reconnu par un automate dans $\mathcal{P}_k$-$\mathsf{AP}^{\vec{\emptyset}}$. D'après la proposition 9.2.5, $\mathcal{M}_{k,n}$ appartient à LANG_k, il existe donc $L_0 = \mathcal{M}_{k,2} \in \mathsf{LANG}_k$ générant le cône rationnel LANG_k.

□

10.3 Jeux à piles itérées

Nous nous intéressons maintenant aux jeux se déroulant dans le graphe de calcul d'un automate à k-piles. Nous prouvons qu'il est possible de décider du gagnant d'un tel jeu et qu'il existe un automate à k-piles "calculant" une stratégie gagnante pour le gagnant. Ces résultats qui étendent ceux de Walukiewicz pour les automates à 1-piles [Wal01] ont déjà été établis de manière efficace dans [Cac03]. Nous proposons ici une autre méthode découlant directement du théorème 9.3.7 énonçant l'équivalence entre les automates à k-piles non-contrôlés et les automates à k-piles contrôlés par des ensembles de piles k-réguliers. Par soucis de simplicité, ces résultats sont établis pour des jeux se déroulant dans le graphe de calcul d'un automate non

contrôlé mais nous obtenons exactement les mêmes résultats si nous considérons des automates contrôlés par des propriétés SOM-définissables.

Les définitions et principaux résultats portant sur les jeux de parité ont été introduits dans le chapitre §4.3.1.

Définition 10.3.1 (Jeu à piles itérées). *Un **jeu à k-piles** est la donnée d'un automate à k-piles sans entrée $\mathcal{A} = (Q, \emptyset, (A_1, \ldots, A_k), \emptyset, \Delta, q_0, \bot)$, d'une partition de Q en deux ensembles Q_0 et Q_1 et d'une application de Q vers un sous-ensemble borné de $\mathbb{N}$, $c : Q \to [0, max]$. La donnée de ces trois objets définit le jeu de parité*

$$J(\mathcal{M}, Q_0, Q_1, c) = (V_0, V_1, E, c, (q_0, \bot))$$

où $V_\epsilon = Q_\epsilon \times k\text{-Pile}(A_1, \ldots, A_k)$, $\epsilon \in \{0, 1\}$, et E contient toutes les couples d'états totaux $((p, \omega), (q, \omega))$ tels que $(p, \omega) \rightarrowtriangle_{\mathcal{M}} (q, \omega')$ et c est étendue en une application de V dans $[0, max]$ en posant pour tout $(p, \omega) \in V$, $c(p, \omega) = c(p)$. En d'autres mots, $(V_0 \cup V_1, E)$ est le graphe de calcul de $\mathcal{A}$.

Une stratégie positionnelle (pour le joueur 0) est une application $\mathrm{s} : V_0 \to \Delta$. Si $\delta = (p, w, \vec{o}, \mathrm{instr}, q)$, alors $\mathrm{s}(p, \omega) = \delta$ signifie qu'en position (p, ω) le joueur 0 doit (et peut) se déplacer en position $(q, \mathrm{instr}(\omega))$.

Comme précédemment dans la section 4.3.2, il sera souvent utile de voir une stratégie positionnelle comme un vecteur de sous-ensembles de k-piles indicés par les transitions de l'automate définissant l'arène du jeu.

Définition 10.3.2 (Stratégie positionelle pour jeux à piles). *Soit $J = (\mathcal{A}, Q_0, Q_1, c)$ un jeu à k-piles avec $\mathcal{A} = (Q, (A_1, \ldots, A_k), \emptyset, \Delta, q_0, F)$ et $\Delta = \{\delta_1, \ldots, \delta_d\}$. (Nous dirons alors que le jeu est de taille d).*

Toute stratégie positionnelle (pour le joueur 0) s est représentée par un vecteur $\vec{S}_\mathrm{s} = (S_{\mathrm{s},1}, \ldots, S_{\mathrm{s},d})$ avec $S_{\mathrm{s},i} = \{\omega \mid \mathrm{s}(p, \omega) = \delta_i\}$.

10.3.1 Décider du gagnant d'un jeu à k-piles

Étant donné un jeu à k-piles J, nous montrons que le problème *il existe une stratégie gagnante pour 0* se réduit à la satisfiabilité d'une formule close dans la structure $\mathbf{Pile_k}$ correspondant au graphe des instructions classiques. Nous montrons également qu'il existe une formule dans $\mathbf{Pile_k}$ dont les modèles sont exactement les vecteurs $\vec{S}_\mathrm{s}$ tels que s est une stratégie gagnante.

Nous procédons par étapes, en montrant d'abord que pour toute stratégie pour 0 sans mémoire s, le problème de savoir si s est gagnante se réduit à la satisfiabilité d'une SOM-formule close dans le graphe de calcul de l'automate définissant l'arène du jeu, enrichie de relations codant la stratégie.

La plupart des méthodes utilisées ici sont similaires à celles appliquées dans le chapitre 4.3.2.

Afin de pouvoir tester la priorité des sommets, nous modifions l'automate d'origine de façon à ce que les arcs de son graphe de calcul soient étiquetés par la transition appliquée.

Définition 10.3.3. *Soit $\mathcal{A} = (Q, \emptyset, (A_1, \ldots, A_k), \vec{C}, \Delta, q_0)$ un k-AP sans entrée. Nous notons $\mathcal{A}_\Delta = (Q, \Delta, (A_1, \ldots, A_k), \vec{C}, \Delta', q_0)$ où*

$$\Delta' = \{(p, \delta, w, \mathrm{instr}, q) \mid \delta = (p, w, \vec{o}, \mathrm{instr}, q) \in \Delta\}.$$

Rappelons qu'une formule à m-places sur une signature Sig est une formule sur la signature Sig enrichie de symboles de m relations unaires (voir définition 3.2.5).

Lemme 10.3.4. *Soit $J = J(\mathcal{A}, Q_0, Q_1, c)$ un jeu à k-piles où la relation de transitions de $\mathcal{A}$ est $\Delta = \{\delta_1, \ldots, \delta_d\}$. Il existe une formule close à d-places* EstGagnante *dans* SOM($\mathsf{Graphe}(\mathcal{A}_\Delta)$) *telle que pour toute stratégie sans mémoire* s,

$$\mathsf{Graphe}(\mathcal{A}_\Delta)^{(\mathrm{s}^{-1}(\delta_1),\ldots,\mathrm{s}^{-1}(\delta_d))} \models \textsc{EstGagnante} \text{ ssi } \mathrm{s} \text{ est gagnante.}$$

Preuve : D'après le lemme 4.3.7, il suffit de prouver que la structure $\mathrm{J_s}$ associée au jeu restreint à la stratégie (voir définition 4.3.5) est SOM-définissable dans $\mathsf{Graphe}(\mathcal{A}_\Delta)^{(\mathrm{s}^{-1}(\delta_1),\ldots,\mathrm{s}^{-1}(\delta_d))}$:

- $E_\mathrm{s}((p,\omega),(q,\omega'))$ ssi
  - soit il existe $\delta \in \Delta_1$ tel que $(p,\omega) \overset{\delta}{\rightarrow}_{\mathcal{A}_\Delta} (q,\omega')$,
  - soit il existe $\delta_i \in \Delta_0$ tel que, $(p,\omega) \overset{\delta}{\rightarrow}_{\mathcal{A}^\mathrm{s}} (q,\omega')$ et $(p,\omega) \in \mathrm{s}^{-1}(\delta_i)$,
- le domaine de la structure $\mathrm{J_s}$ est définissable dans $\mathsf{Graphe}(\mathcal{A}_\Delta)^{\vec{\mathrm{s}}}$ comme la clôture de $\{(q_0, \bot_k)\}$ par les relations associées à E_s,
- pour tout $n \in [0, max]$, $c_n(y)$ est exprimable par la formule

$$[y = (q_0, \bot_k) \wedge c(q_0) = n] \vee \bigvee_{\delta=(p,\ldots,q)|c(q)=n} \exists x,\ E_\delta(x,y).$$

Donc id : $\mathrm{J_s} \rightarrow \mathsf{Graphe}(\mathcal{A}_\Delta)^{(\mathrm{s}^{-1}(\delta_1),\ldots,\mathrm{s}^{-1}(\delta_d))}$ est une SOM-interprétation, ce qui prouve le lemme.

□

Montrons maintenant que l'ensemble des stratégies gagnantes est "définissable" par une SOM-formule dans le graphe de la machine associée au jeu, c'est à dire qu'il existe une formule dont les modèles dans $\mathsf{Graphe}(\mathcal{A})$ sont exactement les vecteurs $(\mathrm{s}^{-1}(\delta_1), \ldots, \mathrm{s}^{-1}(\delta_d))$ où s est une stratégie positionnelle gagnante.

Proposition 10.3.5. *Pour tout jeu à k-piles $\mathrm{J} = J(\mathcal{A}, Q_0, Q_1, c)$, avec $\Delta_\mathcal{A} = \{\delta_1, \ldots, \delta_d\}$, il existe une* SOM*-formule* $\textsc{reg}(X_1, \ldots, X_d)$ *telle que*

1. $\mathsf{Graphe}(\mathcal{A}_\Delta) \models \exists X_1, \ldots, X_d,\ \textsc{reg}(X_1, \ldots, X_d)$ *ssi* J *est gagnant pour 0,*
2. *pour tous $M_1, \ldots, M_d \subseteq V$,*

$$\mathsf{Graphe}(\mathcal{A}_\Delta) \models \textsc{reg}(M_1, \ldots, M_n)$$

ssi il existe une stratégie positionnelle gagnante s *telle que $\forall i \in [1,d]$, $M_i = \mathrm{s}^{-1}(\delta_i)$.*

Preuve : Soit EstGagnante la formule à d-places définie dans le lemme précédent. En transformant chaque symbole relationnel O_i en place i par une variable X_i n'ayant aucune occurrence dans EstGagnante, nous obtenons une formule EstGagnante' vérifiant pour toute stratégie s :

$\mathsf{Graphe}(\mathcal{A}_\Delta) \models \textsc{EstGagnante'}(\mathrm{s}^{-1}(\delta_1), \ldots, \mathrm{s}^{-1}(\delta_d))$ ssi le jeu est gagnant pour 0.

Posons pour tout $p \in Q_0$, $I_p = \{i \in [1,d] \mid \pi_1(\delta_i) = p\}$.

Un vecteur $(M_1, \ldots, M_d)$ est composé des images inverses d'une stratégie positionnelle ssi

- pour tout $p \in Q_0$, $(M_i)_{i \in I_p}$ est une partition de la restriction de $\mathrm{EtatT}(\mathcal{A}_\Delta)$ aux états totaux ayant pour état p,
- $x \in M_i \implies \exists y,\ x \overset{\delta_i}{\rightarrow}_{\mathcal{A}_\Delta} y$.

Il existe donc une formule $\text{STRAT}(X_1, \ldots, X_d)$ vérifiant
$\mathsf{Graphe}(\mathcal{A}_\Delta) \models \text{STRAT}(M_1, \ldots, M_d)$ ssi $\exists$S positionnelle, $M_i = (\text{S}^{-1}(\delta_i), \forall i \in [1, d]$.

Posons $\text{REG}(X_1, \ldots, X_d) := \text{STRAT}(X_1, \ldots, X_d) \wedge \text{EST}\textsc{Gagnante}'(X_1, \ldots, X_d)$. Cette formule satisfait la proposition.

□

Transférons maintenant ces résultats dans une structure plus générale afin d'avoir un résultat indépendant de l'automate choisit.

Théorème 10.3.6. *Pour tout jeu* J *à* k*-piles de taille* d*, il existe une formule* REG_J *à* d *variables libres, telle que*

1. $\mathbf{Pile_k} \models \exists X_1, \ldots, X_d$, $\text{REG}_J(X_1, \ldots, X_d)$ *ssi 0 gagne le jeu*
2. *pour tous* $S_1, \ldots, S_d \subseteq k$-Pile, $\mathbf{Pile_k} \models \text{REG}_J(S_1, \ldots, S_d)$ *ssi* $S = \{((p, \omega), \delta_i) \mid \omega \in S_i)\}$ *est une stratégie sans mémoire pour le joueur 0 gagnante dans* J*.*

Preuve : Nous procédons ici de façon similaire à la preuve du lemme 4.3.12, en définissant un codage g des éléments de $\text{EtatT}(\mathcal{A})$ par des sous-ensembles de k-Pile.

Supposons que $Q = \{p_1, \ldots, p_\tau\}$, nous posons pour tout $S \subseteq V$, $g(S) = (g_1(S), \ldots, g_\tau(S))$ où $g_i(S) = \{\omega \mid (p_i, \omega) \in S)$, pour $i \in [1, \tau]$.

Par la même méthode que celle utilisée dans la preuve du lemme 4.3.11, nous prouvons que pour toute SOM-formule $\Phi(X_1, \ldots, X_n)$, dans le graphe $\mathsf{Graphe}_k(\mathcal{A})$, il est possible de construire une formule SOM $g(\Phi)(X_{1,1}, \ldots X_{\tau,n}, \ldots, X_{m,1}, \ldots, X_{\tau,n})$ sur $\mathbf{Pile_k}$, telle que

$$\mathsf{Graphe}_k(\mathcal{A}_\Delta) \models \Phi(D_1, \ldots, D_n) \text{ ssi } \mathbf{Pile_k}^{\vec{C}} \models g(\Phi)(g(D_1), \ldots, g(D_n)).$$

En appliquant alors cette construction à la formule reg de la proposition 10.3.5, puis en transformant les symboles de relations associés à la stratégie en variables libres, nous obtenons une formule REG_J satisfaisant le théorème.

□

La proposition 3.3.4 assure que la structure associée aux instructions d'un k-AP admet une théorie SOM décidable. Un corollaire direct du théorème précédent est donc le suivant :

Corollaire 10.3.7. *Pour tout jeu à* k*-piles* J*, les problèmes suivants sont décidables :*

1. *pour une stratégie* S *donnée : "*S *est-elle une stratégie gagnante ?",*
2. *"quel est le gagnant du jeu ?".*

10.3.2 Calculer une stratégie gagnante

Nous avons montré dans la partie II (théorème 6.2.1) que la structure $\mathbf{Pile_k}$ vérifie la propriété MD, qui assure que toute formule satisfiable dans $\mathbf{Pile_k}$ admet un modèle définissable. Donc d'après le théorème 10.3.6, pour tout jeu à k-pile gagnant pour le joueur 0, il existe une stratégie gagnante définissable dans la structure $\mathbf{Pile_k}$. Une stratégie peut être utilisée comme contrôleur d'un k-AP déterministe reconnaissant toutes les parties jouées selon la stratégie, (nous dirons que l'automate "calcule la stratégie"). En appliquant le théorème 9.3.7 permettant de transformer un tel automate en un automate déterministe non contrôlé, nous en déduisons que toute stratégie est calculée par un automate dans k-$\mathsf{AP}^\emptyset$.

Soit $\mathcal{A} = (Q, (A_1, \ldots, A_k), \Delta, q_0, F)$ un k-AP. Une partie $(q_0, \bot_k)(q_1, \omega_1) \cdots (q_n, \omega_n) \cdots$ dans le graphe de $\mathcal{A}_\Delta$ peut être représentée de façon équivalente par le mot infini $\delta_1 \cdots \delta_n \cdots$ sur Δ correspondant à la suite des étiquettes du chemin.

Définition 10.3.8. *Soit J un jeu à k-piles. Un automate $\mathcal{A} \in k$-AP reconnaît une stratégie* s *de* J *si $\mathcal{A}$ est déterministe et* $\mathrm{L}(\mathcal{A})$ *est l'ensemble des parties $\delta_1 \cdots \delta_n \cdots$ jouées selon* s.

Théorème 10.3.9. *Pour tout jeu J à k-piles dont l'arène est le graphe d'un automate dans k-$\mathsf{AP}^{\mathsf{PREC}_k}$, le joueur 0 gagne le jeu ssi le joueur 0 admet une stratégie reconnu par un automate déterministe dans k-$\mathsf{AP}^{\vec{\emptyset}}$.*

Preuve : Soit $J = (\mathcal{A}, Q_0, Q_1, c)$ avec $\mathcal{A} = (Q, (A_1, \ldots, A_k), \Delta, q_0, F) \in k$-AP. D'après le théorème 10.3.6 et puisque $\mathbf{Pile_k}$ satisfait la propriété MD, il existe une stratégie positionnelle s telle $\vec{S}_\mathrm{s}$ est définissable en SOM dans $\mathbf{Pile_k}$. Posons

$$\mathcal{A}^\mathrm{s} = (Q, \Delta, (A_1, \ldots, A_k), \vec{S}_\mathrm{s}, \Delta_\mathrm{s}, q_0, F)$$

où Δ_s est construit en posant pour toute transition $\delta = (p, w, \vec{o}, \mathrm{instr}, q) \in \Delta$:
- si $p \in Q_1$, alors $\{(p, \delta, w, \vec{o}, \mathrm{instr}, q) \mid \vec{o} \in \{0,1\}^d\} \subseteq \Delta_\mathrm{s}$,
- si $p \in Q_0$ alors $\delta_\mathrm{s} = \{(q, \delta, w, \vec{o}, \mathrm{instr}, q) \mid \vec{o} \in \{0,1\}^d, \pi_i(\vec{o}) = 1\} \subseteq \Delta_\mathrm{s}$.

Le graphe de calcul de l'automate $\mathcal{A}^\mathrm{s}$ est un sous-graphe du graphe de $\mathcal{A}$. Le langage reconnu par $\mathcal{A}^\mathrm{s}$ est l'ensemble des parties $\delta_0 \cdots \delta_n \cdots$ dans J jouées selon s.

Cet automate ne contient aucune ε-transition, est déterministe et pour tout $\delta_0 \cdots \delta_n$, tel que $(q_0, \delta_0 \cdots \delta_n, \bot_k) \vdash^*_{\mathcal{M}^\mathrm{s}_\Delta} (q, \varepsilon, \omega)$, alors $\delta_0 \cdots \delta_n$ définit le préfixe d'une partie jouée selon s et aboutissant en (q, ω). Puisque le contrôleur de cet automate appartient à PREC_k, en appliquant maintenant le théorème 9.3.7, nous obtenons un automate déterministe dans k-$\mathsf{Pile}^{\vec{\emptyset}}$ calculant la stratégie s.

□

Remarque 10.3.10. *Nous aurions également pu définir la notion d'automate calculant une stratégie, en utilisant la notion de* transducteur*, c'est à dire, d'automate avec entrées-sorties, comme défini dans [Wal01]. L'automate reçoit en entrée les transitions correspondant aux mouvements du joueur 1, et répond à chaque lecture d'une lettre en entrée, par la sortie de la transition indiquée par la stratégie. Le théorème 9.3.7 s'appliquant également pour un tel automate, nous pouvions établir un analogue au théorème ci-dessus pour cette notion de calcul effectif d'une stratégie.*

Quatrième partie

Suites et automates

Introduction

Dans cette dernière partie, nous étudions les **langages** reconnus par automate à k-piles, et en particulier, les langages définis sur des alphabets composés d'une **unique** lettre. Ces langages peuvent être vus comme des ensembles d'entiers : chaque mot de longueur n est associé bijectivement à l'entier n. Nous définissons une large classe $\Sigma\mathbb{S}_k$ de suites reconnues par k-AP **déterministes** et stable par de nombreuses opérations.

En plus de fournir une large palette d'exemples de langages de niveau k, cette étude se révèle prolifique dans plusieurs domaines Les liens établis dans la deuxième partie entre la logique et les automates nous permettent d'étendre le résultat de Büchi établissant la décidabilité de la théorie du second ordre monadique de la structure $\langle \mathbb{N}, +1 \rangle$ aux structures $\langle \mathbb{N}, +1, s(\mathbb{N}) \rangle$, pour toute suite s dans la classe $\Sigma\mathbb{S}_k$.

Nous définissons également une classe $\mathcal{F}(\mathbb{S}_k)$ de suites de nombres **rationnel** pour laquelle le problème de l'égalité se réduit au problème de l'équivalence entre deux automates à k-piles déterministes. Cette classe bénéficie elle aussi de propriétés de clôture intéressantes et contient certaines classes de suites remarquables. Ces résultats permettent de lier les problèmes algorithmiques sur les suites (voir par exemple [PWZ96]), et les problèmes de décision sur les automates (voir [Sén02]).

Cette partie se compose de quatre chapitres. Le premier est dédié à l'étude des suites d'entiers reconnues par automates, nous donnons des nombreux exemples de base telles que les suites $\mathbb{N}$-rationnelles, ou les solutions d'équation polynomiales, et prouvons la stabilité de cette classe de suites par des opérations simples, ainsi que par des opérations plus élaborées telles que le produit de convolution, la substitution de séries ou encore la composition de suites. Nous étendons dans le chapitre 2, la définition de suites calculables, aux suites doubles (i.e., à deux indices). Ces résultats sont exploités dans le troisième chapitre pour l'étude de propriétés de décidabilité en arithmétique faible. Enfin, nous terminons par l'étude de problèmes algorithmiques sur les suites de nombres rationnels.

Opérateurs sur les suites et séries :

Soit $(\mathbb{Q}, +, \cdot)$ le corps des nombres rationnels. Une *suite* de nombres rationnels est une application $s : \mathbb{N} \to \mathbb{Q}$. Nous notons $s(n)$ (ou parfois s_n) l'image de l'entier n par l'application s. Une suite s peut aussi être vue comme une série formelle :

$$s(X) = \sum_{n=0}^{\infty} s_n X^n.$$

Les opérateurs suivants sont classiques :
E : l'opérateur *décalage*

$$(\mathsf{E}s)(n) = s(n+1),\ \ (\mathsf{E}s)(X) = \frac{s(X) - s(0)}{X}$$

Δ : l'opérateur *différence*

$$(\Delta s)(n) = s(n+1) - s(n),\ \ (\Delta s)(X) = \frac{s(X)(1-X) - s(0)}{X}$$

Σ : l'opérateur *somme généralisée*

$$(\Sigma s)(n) = \sum_{j=0}^{n} s(j),\ \ (\Sigma s)(X) = \frac{s(X)}{1-X}$$

$+$: l'opérateur *somme*

$$(s+t)(n) = s(n) + t(n),\ \ (s+t)(X) = s(X) + t(X)$$

$\cdot$: le *produit externe*, pour tout $r \in \mathbb{Q}$

$$(r \cdot s)(n) = r \cdot s(n)$$

$\odot$: le produit de Hadamard, (ou *produit ordinaire*)

$$(s \odot t)(n) = s(n) \cdot t(n)$$

$\times$: le *produit de convolution*

$$(s \times t)(n) = \sum_{k=0}^{n} s(k) \cdot t(n-k),\ \ (s \times t)(X) = s(X) \cdot t(X)$$

$\circ$: la *composition de suites*

$$(s \circ t)(n) = s(t(n))$$

$^{-1}$: l'opérateur *inverse entier*

$$s^{-1}(n) = |s(\mathbb{N}) \cap [1, \ldots, n]|$$

$\bullet$: la *composition de séries*

$$(s \bullet t)(X) = \sum_{n=0}^{\infty} s(n) \cdot t(X)^n$$

(cette dernière opération est définie dès que $t(0) = 0$, ce qui assure que la famille de séries $(s(n) \cdot t(X)^n)_{n \geq 0}$ est sommable).

L'ensemble de toutes les suites de nombres rationnels est noté $\mathbb{Q}[[X]]$.
La structure $(\mathbb{Q}[[X]], +, \times)$ est un anneau, avec $\mathbb{1} = (1, 0, \ldots, 0, \ldots)$. Soit $r \in \mathbb{Q}$, nous utiliserons la même notation pour les suites (ou séries) :

$$r = r \cdot \mathbb{1} = (r, 0, \ldots, 0, \ldots)$$

tandis que nous utiliserons $\frac{r}{1-X}$ pour la suite (ou série) constante :

$$\frac{r}{1-X} = (r, r, \ldots, r, \ldots).$$

Chapitre 11

Suites d'entiers

Nous définissons une classe de **suites d'entiers** par le biais d'automates à k-piles. Précisément, nous utilisons une sous-classe de k-AP légèrement restrictive, les k-AP à compteur, extension naturelle des 1-AP à compteur dont la mémoire se compose d'un entier naturel.

Les piles des k-AP à compteur contiennent un entier au niveau 1 de la pile : c'est-à-dire que l'alphabet A_1 de niveau 1 est composé d'un unique symbole a_1. Chaque 1-pile apparaissant dans une telle k-pile représente donc l'entier correspondant au nombre d'occurrences de a_1.

Afin d'obtenir une notion de suite *calculée par un k-AP*, nous définissons un schéma standard de calcul. Étant donnée une suite s, nous définissons une "configuration" de départ, dépendant de n, à partir de laquelle $s(n)$ va être calculée. Nous dirons que la suite s est *k-calculable*, si il existe un k-AP à compteur déterministe (et normalisé) $\mathcal{A}$ tel que pour tout $n \geq 0$,

$$(q_0, \alpha^{s(n)}, a_k[a_{k-1}[\cdots a_2[{a_1}^n]\cdots]\ \bot_{k-1}]\ \bot_k) \vdash_{\mathcal{A}}^* (q_0, \varepsilon, \bot_k).$$

La forme simple de pile de départ (qui correspond à ${\varphi_k}^{-1}({a_1}^n a_2 \cdots a_k)$) se prête à la récurrence et à la composition d'automates. Nous montrerons ainsi que la classe des suites ainsi calculables contient de nombreuses suites définies de façon récurrentes telles que les suites $\vec{N}$-rationnelles, les solutions d'équations polynomiales, et est stable par de nombreuses opérations.

Si par exemple t est une autre suite calculée par un k-AP $\mathcal{B}$ à partir de l'état total initial $(q_0, b_k[a_{k-1}[\cdots a_2[{a_1}^n]\cdots]\ \bot_{k-1}]\ \bot_k)$, il est alors facile de construire un nouveau k-AP $\mathcal{C}$ calculant la suite $s+t$ à partir de l'état total $(q_0, c_k[a_{k-1}[\cdots a_2[{a_1}^n]\cdots]\ \bot_{k-1}]\ \bot_k)$.

Il suffit pour cela de faire l'union des transitions de $\mathcal{A}$ et $\mathcal{B}$ avec la transition $(q_0, \varepsilon, c_k a_{k-1} \cdots a_1, q_0, \mathrm{change}_{b_k}\ \mathrm{push}_{a_k})$.

Pour tout $n \geq 0$, $\mathcal{C}$ effectue le calcul suivant :

$$\begin{aligned}
& (q_0, \alpha^{s(n)+t(n)}, c_k[a_{k-1}[\cdots a_2[{a_1}^n]\cdots]\ \bot_{k-1}]\ \bot_k) \\
\vdash_{\mathcal{C}} & (q_0, \alpha^{s(n)+t(n)}, a_k[a_{k-1}[\cdots a_2[{a_1}^n]\cdots]\ \bot_{k-1}] b_k[a_{k-1}[\cdots a_2[{a_1}^n]\cdots]\ \bot_{k-1}]\ \bot_k) \\
\vdash_{\mathcal{C}}^* & (q_0, \alpha^{t(n)}, b_k[a_{k-1}[\cdots a_2[{a_1}^n]\cdots]\ \bot_{k-1}]\ \bot_k) \\
\vdash_{\mathcal{C}}^* & (q_0, \varepsilon, \bot_k).
\end{aligned}$$

Nous verrons au cours de ce chapitre comment procéder pour des opérations plus complexes sur les suites comme le produit de convolution, ou la résolution de systèmes récurrents polynomiaux dont les coefficients sont des suites.

Nous autoriserons également les automates calculant les suites à être contrôlés, mais seulement par des contrôles portant sur le compteur de la 1-pile de tête. Tout les résultats de stabilité sont résumés dans le théorème 11.5.19.

11.1 Automates à compteurs contrôlés

Le choix initial de définir les k-piles en utilisant des fonds de piles permet d'obtenir une bijection avec l'ensemble $\mathcal{P}_k$, toutefois, dans la pratique les fonds de piles alourdissent les notations et rendent difficile la définition d'un produit de concaténation.

Cette dernière partie consistant essentiellement en constructions d'automates, nous choisissons de travailler uniquement avec des piles N-normalisées (voir 1.2.1).

À l'intérieur d'une k-Npile, le dernier atome de chaque occurrence d'une pile de niveau i est $\perp_i$. Il n'est donc pas nécessaire d'écrire les fonds de piles lors de la représentation d'une k-Npile. Considérons par exemple la pile $\omega = a_3[\perp_2]b_3[a_2[a_1\ \perp]\ \perp_2]\ \perp_3 \in$ 3-NPile. Nous pouvons, sans confusion (dès lors que nous connaissons son niveau), écrire $\omega = a_3[\varepsilon]b_3[a_2[a_1]]$ ou même $\omega = a_3b_3[a_2[a_1]]$. Remarquons qu'avec cette notation, nous introduisons une ambiguïté sur le niveau de la pile représentée. Dans l'exemple donné précédemment, on voit clairement que ω est au moins de niveau 3, mais elle peut également être de niveau 4 ou plus. Nous prendrons donc toujours soin de préciser le niveau des piles utilisées.

La même simplification sera appliquée à la liste des symboles de tête. Puisque la liste des symboles de tête de toute pile $\omega \in k$-NPile est toujours de la forme $\mathrm{top}(\omega) = a_k \dots a_i\ \perp^{i-1}$, lorsque le niveau de la pile est connu, nous noterons plutôt $\mathrm{top}(\omega) = a_k \dots a_i$. Ainsi, pour l'exemple précédent, nous préférerons écrire $\mathrm{top}(\omega) = a_3$ plutôt que $\mathrm{top}(\omega) = a_3\ \perp\perp$. De même, dès qu'il n'y a pas de confusion possible sur le niveau de la pile, nous noterons ε plutôt que $\perp_k$.

Pour toute la suite de ce chapitre, nous nous restreindrons à des automates à compteurs (i.e., l'alphabet de niveau 1 ne contient qu'une lettre), dont le contrôle porte uniquement sur la valeur du compteur de tête, et N-normalisés (voir §2.3.1).

Définition 11.1.1 (Automates à compteurs normalisés). *Soit $\vec{N}$ un vecteur d'ensembles d'entiers naturels. Nous notons k-AC$^{\vec{N}}$ l'union des k-NAP$^{(|\vec{N}|_{A_1}{}^{-1})_1}(A_1, \dots, A_k)$, pour tous alphabets $A_1, \dots, A_k$ tels que $|A_1| = 1$. L'union de tous les k-AC$^{\vec{N}}$ est notée k-AC, la classe des k-AC$^{\vec{N}}$ déterministes est notée k-ACD$^{\vec{N}}$.*

Dans la pratique, nous utiliserons souvent des automates dans ${}^*k$-NAP (voir définition 7.2.3), ayant les mêmes propriétés. Rappelons que d'après le lemme 7.2.4, tout automate dans ${}^*k$-NAP est équivalent (par simulation déterministe), à un automate dans k-NAP.

11.2 Suites définies par automates

Nous formalisons maintenant la notion de suite d'entiers calculable par automate à compteurs et énonçons quelques propriétés de ces suites.

Définition 11.2.1 (Suites calculables). *Soient $k \geq 1$ et $\vec{N}$ un vecteur de sous-ensembles de $\mathbb{N}$. Une suite d'entiers naturels s est dite $(k, \vec{N})$-**calculable** si il existe $\mathcal{A} \in k$-ACD$^{\vec{N}}$ défini sur les alphabets de piles $A_1, \dots, A_k$ et tel que $\forall i \geq 1$, chaque A_i contient une lettre a_i et :*

($\mathbf{H1}(a_k \cdots a_1)$) $\forall n \geq 0,\ (q_0, \alpha^{s(n)}, a_k[a_{k-1}[\ldots[a_2[{a_1}^n]]\ldots]]) \vdash^*_{\mathcal{A}} (q_0, \varepsilon, \varepsilon)$,
($\mathbf{H2}$) $\forall q \in Q,\ \Delta_{\mathcal{A}}$ *ne contient pas de membre gauche de la forme* (q, α, ε) *ou* $(q, \varepsilon, \varepsilon)$.

Nous notons ${\mathbb{S}_k}^{\vec{N}}$ *l'ensemble des suites* $(k, \vec{N})$*-calculables. Lorsque* $\vec{N} = \vec{\emptyset}$*, nous dirons simplement que* s *est* k*-calculable et nous noterons* $\mathbb{S}_k$ *plutôt que* $\mathbb{S}_k^{\vec{\emptyset}}$.

Ce schéma de calcul va permettre de définir de nombreuses récurrences. La principe est exposé ici pour l'exemple le plus simple :

Exemple 11.2.2 (Récurrence linéaire). *Considérons la suite* s *définie par la récurrence suivante,*

$$s(0) = 2;\ \ \forall n \geq 0,\ s(n+1) = 2s(n) + 1.$$

Nous construisons un automate déterministe $\mathcal{A} \in$ 2-AC *calculant* s*, vérifiant* ($\mathbf{H1}(a_2a_1)$) *et* ($\mathbf{H2}$).

La clé de la construction est de trouver les transitions permettant les calculs suivants :

1. $(q_0, \alpha^{s(0)}, a_2[\varepsilon]) \vdash^*_{\mathcal{A}} (q_0, \varepsilon, \varepsilon)$,
2. $\forall n \geq 0,\ \forall \omega \in$ 2-NPile, $(q_0, \varepsilon, a_2[{a_1}^{n+1}]\omega) \vdash^*_{\mathcal{A}} (q_0, \varepsilon, b_2[{a_1}^n]a_2[{a_1}^n]a_2[{a_1}^n]\omega)$,
3. $\forall n \geq 0,\ \forall \omega \in$ 2-NPile, $(q_0, \alpha, b_2[{a_1}^n]\omega) \vdash^*_{\mathcal{A}} (q_0, \varepsilon, \omega)$.

Vérifions par induction sur $n \geq 0$ *que tout automate réalisant ces calculs vérifie la propriété* $\mathbf{P}(n)$ *suivante :* $\forall \omega \in$ 2-NPile,

$$(q_0, \alpha^{s(n)}, a_2[{a_1}^n]\omega) \vdash^*_{\mathcal{A}} (q_0, \varepsilon, \omega).$$

L'hypothèse (1) prouve $\mathbf{P}(0)$*. Supposons* $\mathbf{P}(n)$ *pour* $n \geq 0$*. Pour tout* $\omega \in$ 2-NPile*, nous obtenons en appliquant l'hypothèse (2), l'hypothèse (3), puis deux fois* $\mathbf{P}(n)$ *:*

$$\begin{array}{lll}
(q_0, \alpha^{s(n+1)}, a_2[{a_1}^{n+1}]\omega) & \vdash^*_{\mathcal{A}} & (q_0, \alpha^{s(n+1)}, b_2[{a_1}^n]a_2[{a_1}^n]a_2[{a_1}^n]\omega) \\
& \vdash^*_{\mathcal{A}} & (q_0, \alpha^{s(n+1)-1}, a_2[{a_1}^n]a_2[{a_1}^n]\omega) \\
& \vdash^*_{\mathcal{A}} & (q_0, \alpha^{s(n+1)-s(n)-1}, a_2[{a_1}^n]\omega) \\
& \vdash^*_{\mathcal{A}} & (q_0, \varepsilon, \omega).
\end{array}$$

La propriété $\mathbf{P}(n)$ *est donc vraie pour tout* $n \geq 0$*, et dans le cas particulier où* $\omega = \varepsilon$*, l'automate* $\mathcal{A}$ *vérifie* ($\mathbf{H1}(a_2a_1)$).

Il nous faut maintenant montrer que nous savons trouver un automate dans 2-ACD *réalisant les calculs (1), (2) et (3), ainsi que l'hypothèse* $\mathbf{H2}$.

Posons $\mathcal{A} = (\{q_0\}, \{\alpha\}, (A_1, A_2), \Delta, q_0)$ *où* $A_1 = \{a_1\}$*,* $A_2 = \{a_2, b_2\}$ *et :*

(a) $\Delta(q_0, \varepsilon, a_2) = (\mathrm{change}_{b_2}\, \mathrm{push}_{b_2}, q_0)$,
(b) $\Delta(q_0, \varepsilon, a_2a_1) = (\mathrm{pop}_1\, \mathrm{push}_{a_2}\, \mathrm{push}_{b_2}, q_0)$,
(c) $\delta(q_0, \alpha, b_2) = \delta(q_0, \alpha, b_2a_1) = (\mathrm{pop}_2, q_0)$.

Cet automate est déterministe et vérifie $\mathbf{H2}$*. Les transitions (a) et (c) permettent le calcul donné dans l'hypothèse 1, la transition (b) rend le calcul (2) valide, tandis que la transition (c) permet le calcul (3).*

Voici le calcul détaillé de $s(2) = 11$:

$$\begin{aligned}(q_0, \alpha^{11}, a_2[a_1a_1]) &\vdash_{\mathcal{A}} (q_0, \alpha^{11}, b_2[a_1]a_2[a_1]a_2[a_1]) \vdash_{\mathcal{A}} (q_0, \alpha^{10}, a_2[a_1]a_2[a_1]) \\ &\vdash_{\mathcal{A}} (q_0, \alpha^{10}, b_2a_2a_2a_2[a_1]) \vdash_{\mathcal{A}} (q_0, \alpha^{9}, a_2a_2a_2[a_1]) \\ &\vdash_{\mathcal{A}} (q_0, \alpha^{9}, b_2b_2a_2a_2[a_1]) \vdash^2_{\mathcal{A}} (q_0, \alpha^{7}, a_2a_2[a_1]) \\ &\vdash_{\mathcal{A}} (q_0, \alpha^{7}, b_2b_2a_2[a_1]) \vdash^2_{\mathcal{A}} (q_0, \alpha^{5}, a_2[a_1]) \\ &\vdash_{\mathcal{A}} (q_0, \alpha^{5}, b_2a_2a_2) \vdash_{\mathcal{A}} (q_0, \alpha^{4}, a_2a_2) \\ &\vdash_{\mathcal{A}} (q_0, \alpha^{4}, b_2b_2a_2) \vdash^2_{\mathcal{A}} (q_0, \alpha^{2}, a_2) \\ &\vdash_{\mathcal{A}} (q_0, \alpha^{2}, b_2b_2) \vdash^2_{\mathcal{A}} (q_0, \varepsilon, \varepsilon).\end{aligned}$$

À cause de la condition **H1**, l'automate associé à une suite k-calculable reconnaît toujours le langage vide. Par contre, en l'enrichissant de transitions bien choisies, il est possible de reconnaître différents langages dépendant de la suite calculée. Voyons quelques langages reconnaissables par des extensions de l'automate donné dans l'exemple précédent.

Exemple 11.2.3. *Soit $\mathcal{A} = (\{q_0\}, \{\alpha\}, (A_1, A_2), \Delta, q_0)$ où $A_1 = \{a_1\}$, $A_2 = \{a_2, b_2\}$, l'automate donné dans l'exemple 11.2.2, nous construisons deux nouveaux automates $\mathcal{A}_1$ et $\mathcal{A}_2$.*

1. *Posons $\mathcal{A}_1 = (\{p_0, q_0, q_F\}, \{\alpha\}, (A_1, A_2), \Delta \cup \Delta_1, p_0, \{q_F\})$ avec*
   - $\Delta_1(p_0, \varepsilon, \varepsilon) = (\text{push}_{a_2}, p_0)$
   - $\Delta_1(p_0, \varepsilon, a_2) = \Delta(p_0, \varepsilon, a_2a_1) = \{(\text{stay}, q_0), (\text{push}_{a_1}, p_0)\}$
   - $\Delta_1(p_0, \varepsilon, \varepsilon) = (\text{stay}, q_F)$.

   Ce nouvel automate est non-déterministe et reconnaît le langage $L_1(s) = \{\alpha^{s(n)} \mid n \geq 0\}$. En effet, tout calcul de $\mathcal{A}_1$ est de la forme suivante :

$$(p_0, \varepsilon, \varepsilon) \vdash^*_{\mathcal{A}_1} (p_0, \varepsilon, a_2[{a_1}^n]) \vdash_{\mathcal{A}_1} (q_0, \varepsilon, a_2[{a_1}^n]) \vdash^*_{\mathcal{A}_1} (q_0, \varepsilon, \varepsilon) \vdash_{\mathcal{A}_1} (q_F, \varepsilon, \varepsilon).$$

2. *Posons $\mathcal{A}_2 = (\{q_0, q_F\}, \{\alpha\}, (A_1, A_2 \cup \{b_2\}), \Delta \cup \Delta_2, q_0, \{q_0\})$ avec*
   - $\Delta_2(q_0, \varepsilon, \varepsilon) = (\text{push}_{b_2}\, \text{push}_{a_2}, q_0)$,
   - $\Delta_2(q_0, \varepsilon, b_2) = \Delta_2(q_0, \varepsilon, b_2a_1) = (\text{push}_{a_1}, q_F)$,
   - $\Delta_2(q_F, \varepsilon, b_2a_1) = (\text{push}_{a_2}, q_0)$.

   Cet automate est déterministe et reconnaît le langage $L_2(s) = \{\alpha^{\Sigma s(n)} \mid n \geq 0\}$ (rappelons que $\Sigma s(n) = \sum_{m \in [0,n]} s(m)$).

   En effet, nous pouvons vérifier par une induction évidente que pour tout $n \geq 0$,

$$(q_0, \alpha^{\Sigma s(n)}, \varepsilon) \vdash^*_{\mathcal{A}_2} (q_0, \varepsilon, b_2[{a_1}^n]) \vdash_{\mathcal{A}_2} (q_F, \varepsilon, b_2[{a_1}^{n+1}]) \vdash_{\mathcal{A}_2} (q_0, \varepsilon, a_2[{a_1}^{n+1}]b_2[{a_1}^{n+1}]) \cdots$$

Ces constructions peuvent être appliquées à toute suite calculable. Nous montrons dans les lemmes suivants qu'à partir d'un k-ACD$^{\vec{N}}$ calculant une suite s au sens de la définition 11.2.1, il est possible de construire un k-AC$^{\vec{N}}$ reconnaissant le langage $L_1 = \{\alpha^{s(n)}, n \geq 0\}$ et un k-ACD$^{\vec{N}}$ reconnaissant le langage $L_2 = \{\alpha^{\Sigma s(n)}\}_{n \in \mathbb{N}}$.

Lemme 11.2.4. *Pour toute suite $s \in \mathbb{S}_k^{\vec{N}}$, il existe un automate non déterministe $\mathcal{A}_1 \in k$-AC$^{\vec{N}}$ tel que $\mathrm{L}(\mathcal{A}_1) = \{\alpha^{s(n)} \mid n \geq 0\}$.*

Preuve : Supposons la suite s calculée par $\mathcal{A} = (Q, \{\alpha\}, (A_1, \ldots, A_k), \vec{N}, \Delta, q_0) \in k$-ACD vérifiant (**H1**$(a_k \cdots a_1)$) et (**H2**). Posons

$$\mathcal{A}_1 = (Q \cup \{p_0, q_F\}, \{\alpha\}, (A_1, \ldots, A_k \cup \{b_k\}), \vec{N}, \Delta \cup \Delta_1, p_0, \{q_F\})$$

avec pour tout $\vec{o} \in \{0,1\}^m$:

- $\Delta_1(p_0, \varepsilon, \varepsilon, \vec{o}) = (\text{push}_{a_k} \text{push}_{a_{k-1}} \cdots \text{push}_{a_2}, p_0)$,
- $\Delta_1(p_0, \varepsilon, a_k a_{k-1} \cdots a_2, \vec{o}) = \Delta_1(p_0, \varepsilon, a_k a_{k-1} \cdots a_2 a_1, \vec{o}) = \{(\text{stay}, q_0), (\text{push}_{a_1}, p_0)\}$
- $\Delta_1(p_0, \varepsilon, \varepsilon) = (\text{stay}, q_F)$.

Alors, chaque calcul de l'automate est de la forme :

$$\begin{array}{lll} (p_0, \alpha^{s(n)}, \varepsilon) & \vdash_{\mathcal{A}_1} & (p_0, \alpha^{s(n)}, a_k[a_{k-1} \cdots [a_2[{a_1}^n]] \cdots]) \\ & \vdash_{\mathcal{A}_1} & (q_0, \alpha^{s(n)}, a_k[a_{k-1} \cdots [a_2[{a_1}^n]] \cdots]) \\ & \vdash^*_{\mathcal{A}_1} & (q_0, \varepsilon, \varepsilon) \vdash_{\mathcal{A}_1} (q_F, \varepsilon, \varepsilon). \end{array}$$

□

Lemme 11.2.5. *Pour toute suite $s \in \mathbb{S}_k^{\vec{N}}$, il existe $\mathcal{A}_2 \in k\text{-ACD}^{\vec{N}}$ tel que $\mathrm{L}(\mathcal{A}_2) = \{\alpha^{\Sigma s(n)} \mid n \in \mathbb{N}\}$ et $\mathsf{ACC}_(\mathcal{A}_2) = \{a_1^n \mid n \geq 0\}$.*

Preuve : Supposons que s est calculée par $\mathcal{A} = (Q, \{\alpha\}, (A_1, \ldots, A_k), \vec{N}, \Delta, q_0, F) \in k\text{-ACD}$ vérifiant (**H1**$(a_k \cdots a_1)$) et (**H2**). Posons

$$\mathcal{A}_2 = (Q \cup \{q_\perp\}, \{\alpha\}, (A_1, \ldots, A_k \cup \{b_k\}), \vec{N}, \Delta \cup \Delta_2, q_0, \{q_\perp\})$$

où Δ_2 est l'union pour $\vec{o} \in \{0,1\}^{|\vec{N}|}$ des transitions suivantes :

- $\Delta_2(q_0, \varepsilon, \varepsilon, \vec{o}) = (\text{push}_{b_k} \text{push}_{a_{k-1}} \cdots \text{push}_{a_2} \text{push}_{a_k}, q_0)$,
- $\Delta_2(q_0, \varepsilon, b_k a_{k-1} \cdots a_2, \vec{o}) = \Delta_2(q_0, \varepsilon, b_k a_{k-1} \cdots a_1, \vec{o}) = (\text{push}_{a_1}, q_F)$,
- $\Delta_2(q_F, \varepsilon, b_k a_{k-1} \cdots a_1, \vec{o}) = (\text{push}_{a_k}, q_0)$.

Ces transitions sont toutes incompatibles deux à deux. De plus, comme $\mathcal{A}$ vérifie (**H2**), b_k est un nouveau symbole et q_F est un nouvel état, chaque transition de Δ_2 est incompatible avec toute transition de Δ. L'automate $\mathcal{A}_2$ est donc déterministe.

Posons $\omega_n = b_k[a_{k-1} \cdots [a_2[{a_1}^n]] \cdots]$, une induction évidente permet de prouver que pour tout $n \geq 0$

$$(q_0, \alpha^{\Sigma s(n)}, \varepsilon) \vdash^*_{\mathcal{A}_2} (q_F, \varepsilon, b_k[a_{k-1} \cdots [a_2[{a_1}^{n+1}]] \cdots]).$$

□

Toujours sur le même mode, il est également possible de reconnaître d'autres langages.

Lemme 11.2.6. *Pour toute suite $s \in \mathbb{S}_k^{\vec{N}}$, chacun des langages*

$$L_3 = \{\alpha^{s(0)} \beta \alpha^{s(1)} \beta \cdots \alpha^{s(n)} \beta \mid n \geq 0\} \text{ et } L_4 = \{\alpha^{s(0)} \beta \alpha^{s(1)} \beta^2 \cdots \beta^n \alpha^{s(n)} \mid n \geq 0\}$$

est reconnu par un automate dans $k\text{-ACD}^{\vec{N}}$.

Proposition 11.2.7. *Pour toutes suites $s \in \mathbb{S}_\ell$, et $t \in \mathbb{S}_k^{\Sigma s(\mathbb{N})}$,*

1. *le langage $\{\alpha^{\Sigma t(n)} \mid n \geq 0\}$ est reconnu par un automate* **déterministe** *dans $(k+\ell)\text{-AP}$,*
2. *le langage $\{\alpha^{t(n)} \mid n \geq 0\}$ est reconnu par un automate dans $(k+\ell)\text{-AP}$.*

Preuve : D'après le lemme 11.2.5, $\{{a_1}^{\Sigma s(n)} \mid n \in \mathbb{N}\} \in \mathsf{DLANG}_l$. Tout automate dans $k\text{-AC}^{\Sigma s(\mathbb{N})}$ est par définition un automate dans $k\text{-AP}^{(\{{a_1}^{\Sigma s(n)} \mid n \in \mathbb{N}\})_1}$, le théorème 9.3.11 implique donc que pour tout automate $\mathcal{A} \in k\text{-AC}^{\Sigma s(\mathbb{N})}$, il existe une simulation déterministe de $\mathcal{A}$ par un automate dans $(k+l)\text{-AP}$. En particulier,

1. d'après le lemme 11.2.5, $\{\alpha^{\Sigma t(n)} \mid n \geq 0\}$ est reconnu par un automate *déterministe* dans k-$\mathsf{AC}^{\Sigma s(\mathbb{N})}$, et donc également par un $(k+\ell)$-AP *déterministe*,
2. d'après le lemme 11.2.4, le langage $\{\alpha^{t(n)} \mid n \geq 0\}$ est reconnu par un automate dans k-$\mathsf{AC}^{\Sigma s(\mathbb{N})}$, et donc également par un automate dans $(k+\ell)$-AP.

□

11.3 Boite à outils

Nous introduisons maintenant les différents outils qui nous permettront de formaliser les preuves à venir. Nous associons tout d'abord à chaque automate une "grammaire" correspondant à la notion de grammaire *hors-contexte* sur un alphabet infini. Nous introduisons également la notion de terme itéré qui permet d'introduire des variables appelée "indéterminées" aux sein des k-piles. Ceci permettra de définir des schémas généraux de calcul par un automate.

11.3.1 Dérivation

Nous associons à $\mathcal{A}$ un "alphabet" infini

$$V_{\mathcal{A}} = \{(p, \omega, q) \mid p, q \in Q,\ \omega \in k\text{-}\mathsf{NPile} - \{\bot_k\}\} \tag{16}$$

et un ensemble de *productions* noté $P_{\mathcal{A}}$ et composé des règles suivantes :
– les règles de *transition* :

$$(p, \omega, q) \rightarrow_{\mathcal{A}} \alpha_\varepsilon(p', \omega', q)$$

si $(p, \alpha_\varepsilon, \omega) \vdash_{\mathcal{A}} (p', \varepsilon, \omega')$ et $q \in Q$ est arbitraire,

$$(p, \omega, q) \rightarrow_{\mathcal{A}} \alpha_\varepsilon$$

si $(p, \alpha_\varepsilon, \omega) \vdash_{\mathcal{A}} (q, \varepsilon, \varepsilon)$.
– les règles de *décomposition* :

$$(p, \omega, q) \rightarrow_{\mathcal{A}} (p, \eta, r)(r, \eta', q)$$

si $\omega = \eta \cdot \eta', \eta \neq \varepsilon, \eta' \neq \varepsilon$ et $r \in Q$ est arbitraire.

La relation *dérivation simple* générée par $\mathcal{A}$ et notée $\rightarrow_{\mathcal{A}}$, est le plus petit sous-ensemble de $(V \cup \Sigma)^* \times (V \cup \Sigma)^*$ contenant $P_{\mathcal{A}}$ et compatible avec le produit à gauche et le produit à droite. Finalement, la relation *dérivation* générée par $\mathcal{A}$ est notée $\rightarrow^*_{\mathcal{A}}$ et correspond à la clôture réflexive et transitive de $\rightarrow_{\mathcal{A}}$. Ces définitions correspondent à la notion usuelle de *grammaire hors-contexte* associée à l'automate à 1-piles $\mathcal{A}_1$ suivant.

L'alphabet de pile est $A = \{a[\omega] \mid a \in A_k,\ \omega \in (k-1)\text{-}\mathsf{NPile}\}$ et la relation de transition est

$$\Delta_1(q, \alpha_\varepsilon, a[\omega]) = \{(\eta', q') \in Q \times \Gamma_1^* \mid (q, \alpha, a[\omega]) \vdash_{\mathcal{A}} (q', \varepsilon, \eta')\}.$$

Bien sûr, dès que $k \geq 2$, l'alphabet A est infini, mais toutes les propriétés usuelles de la relation $\rightarrow_{\mathcal{A}} = \rightarrow_{\mathcal{A}_1}$ et ces liens avec $\vdash_{\mathcal{A}} = \vdash_{\mathcal{A}_1}$ restent vrais dans ce contexte (voir [Har78, preuve du théorème 5.4.3, pages 151-158]). En particulier, pour tout $\sigma \in \Sigma^*, p, q \in Q, \omega \in A^*$

$$(p, \omega, q) \rightarrow^*_{\mathcal{A}} \sigma \Leftrightarrow (p, \sigma, \omega) \vdash^*_{\mathcal{A}} (q, \varepsilon, \varepsilon).$$

La notion de dérivation permettra de décomposer proprement les calculs d'un automate. Nous utiliserons fréquemment le lemme suivant :

Lemme 11.3.1. *Soient $p_i, q_i \in Q$, $\omega_i \in A^*$ pour $i \in \{1,2,3\}$. Les propriétés suivantes sont équivalentes :*

1. $(p_1,\omega_1,q_1) \rightarrow_{\mathcal{A}}^* (p_2,\omega_2,q_2)(p_3,\omega_3,q_3)$
2. *il existe $\omega_2', \omega_3' \in A^*$, tels que :*

$$(p_1,\varepsilon,\omega_1) \vdash_{\mathcal{A}}^* (p_2,\varepsilon,\omega_2\omega_2'); \quad (q_2,\varepsilon,\omega_2') \vdash_{\mathcal{A}}^* (p_3,\varepsilon,\omega_3\omega_3'); \quad (q_3,\varepsilon,\omega_3') \vdash_{\mathcal{A}}^* (q_1,\varepsilon,\varepsilon).$$

Nous supposerons généralement que A_k et Q sont disjoints pour tout $k \geq 1$, nous pourrons ainsi, sans confusion possible, omettre les virgules dans (p,ω,q).

11.3.2 Termes

Nous définissons ici une notion de *termes itérés*, proche de celle de k-Npile et permettant d'introduire les notions d'indéterminées et de substitution.

Nous fixons une famille $(\mathcal{I}_k)_{k\geq 0}$ d'ensembles de symboles : $\mathcal{I}_k = \{\Omega, \Omega', \Omega'', \ldots, \Omega_1, \Omega_2, \ldots\}$ qui désigne l'ensemble des **indéterminées** de niveau k. Nous supposons que les $\mathcal{I}_k$ sont tous disjoints deux à deux et que tous les alphabets de piles $A_1, \ldots, A_k, \ldots$ que nous considérerons sont tous disjoints de tous les $\mathcal{I}_j$. Un k-terme est une k-pile N-normalisée dans laquelle sont ajoutés des symboles n'appartenant pas aux alphabets de piles. Chaque indéterminée de niveau i (i.e., dans $\mathcal{I}_i$) peut être placée n'importe où (avant $\perp$) au niveau i d'un terme. Définissons l'ensemble k-Terme$(A_1, \ldots, A_k)$ des termes de niveau k par induction sur $k \geq 0$:

- 0-Terme $= \perp$
- $(k+1)$-Terme$(A_1, \ldots, A_{k+1}) = (A_{k+1}[k\text{-Terme}(A_1, \ldots, A_k)] \cup \mathcal{I}_{k+1})^* \perp_{k+1}$.

Comme pour les N-piles, nous n'écrirons pas les symboles de fond de piles lors de la représentation d'un terme.

Un k-terme T sera noté $T[\Omega_1, \ldots, \Omega_n]$ pour signifier que les seules indéterminées occurrentes dans T sont $\Omega_1, \ldots, \Omega_n$.

Le produit de concaténation sur k-NPile se généralise naturellement aux éléments de k-Terme. Il en est de même de l'opération top et des instructions push, pop et change.

Pour tout $k \geq 0$, $\text{top}(\perp_k) = \perp^k$ (c'est à dire $\text{top}(\varepsilon) = \varepsilon$) et pour tout $T \in (k+1)$-Terme, si $T = \Omega \cdot \omega$ alors $\text{top}(T) = \Omega$, et si $T = a[T_1] \cdot T'$, alors $\text{top}(T) = a \cdot \text{top}(T_1)$.

Pour tout terme T tel que $\text{top}_i(T)$ est une indéterminée, les instructions de niveau i push_{a_i}, pop_i et change_i sont indéfinies, sinon, elles sont définies comme sur les k-piles.

11.3.3 Substitutions

Étant donnés $T[\Omega_1, \ldots, \Omega_n] \in k$-Terme$(A_1, \ldots, A_k)$ avec $\Omega_i \in \mathcal{I}_{k_i}$ pour $i \in [1,n]$, $k_i \in [1,k]$ et $T_1 \in k_1$-Terme$, \ldots, T_n \in k_n$-Terme, nous désignons par $T[T_1, \ldots, T_n]$ le k-terme obtenu en substituant T_i à Ω_i.

Il est également possible de substituer les indéterminées par des k_i-piles :

si pour tout $i \in [1,n]$, $\omega_i \in k_i$-NPile, alors $T[\omega_1, \ldots, \omega_n]$ désigne la k-Npile obtenue en substituant ω_i à Ω_i. Remarquons que ces substitutions sont rendues possibles par l'existence du produit de concaténation sur les Npiles et les termes.

Nous énonçons maintenant un "principe de substitution" qui sera souvent utilisé au cours des preuves. Il permettra de définir des schémas "types" de calcul et de dérivation dans un automate.

Étant donné $\mathcal{A} \in k\text{-}\mathsf{AC}^{\vec{N}}$, nous étendons naturellement les relations $\vdash_{\mathcal{A}}$ et $\rightarrow \mathcal{A}$ aux termes n'ayant **aucune indéterminée de niveau 1** (pour que les tests soient indépendants des futures substitutions des indéterminées).

Lemme 11.3.2. *Étant donné $\mathcal{A} \in k\text{-}\mathsf{AC}^{\vec{N}}$ et $\vec{\Omega} = (\Omega_1, \ldots, \Omega_n)$ où chaque Ω_i est une indéterminée de niveau $k_i \in [2, k]$. Si $T[\vec{\Omega}]$ et $T'[\vec{\Omega}]$ sont deux éléments de k-$\mathsf{Terme}(A_1, \ldots, A_k)$, alors pour tous p, q, p', $q' \in Q$,*

$$\textit{si } (pT[\vec{\Omega}]q) \rightarrow^*_{\mathcal{A}} (p'T'[\vec{\Omega}]q'), \textit{ alors}$$

- *pour tout $\vec{H} = (H_1, \ldots, H_n)$ tel que pour tout $i \in [1, n]$, H_i est un k_i-terme,*

$$(pT[\vec{H}]q) \rightarrow^*_{\mathcal{A}} (p'T'[\vec{H}]q'),$$

- *pour tout $\vec{\omega} = (\omega_1, \ldots, \omega_n)$ tel que pour tout $i \in [1, n]$, ω_i est une k_i-Npile,*

$$(pT[\vec{\omega}]q) \rightarrow^*_{\mathcal{A}} (p'T'[\vec{\omega}]q').$$

La preuve de ce lemme est immédiate. L'idée clé est que comme $A_i \cap \mathcal{I}_i = \emptyset$ $\forall i \geq 1$, les symboles Ω_i peuvent être copiés ou effacés durant une dérivation mais ne peuvent pas influer sur la suite de règles utilisées durant une dérivation puisque l'automate s'arrête dès que le liste des symboles de tête contient une indéterminée.

11.4 Quelques suites calculables

Nous montrons dans cette section que de nombreuses suites définies par des relations récursives sont calculables par automates à k-piles.

Nous prouvons tout d'abord que les suites $\mathbb{N}$-rationnelles sont toutes 2-calculables. Une suite $\mathbb{N}$-rationnelle est la solution d'un systèmes d'équations linéaires récursives, comme par exemple la suite de Fibonacci.

Nous montrons ensuite que les suites solutions d'une équation récurrente polynomiale à coefficients entiers sont 3-calculables.

Enfin, nous montrons que l'inverse entier d'une suite strictement croissante s est $(k, s(\mathbb{N}))$-calculable.

Définition 11.4.1 (Suites $\mathbb{N}$-rationnelles). *Une suite $(u_n)_{n\geq 0}$ est $\mathbb{N}$-rationnelle ssi il existe une matrice M dans $\mathbb{N}^{d\times d}$ et deux vecteurs L dans $\mathbb{B}^{1\times d}$ et C dans $\mathbb{B}^{d\times 1}$ tels que $u_n = L \cdot M^n \cdot C$.*

Une définition équivalente est de considérer un graphe fini G, avec arcs multiples et des ensembles I et F de sommets tels que M est la matrice d'adjacence de G et L et C sont les vecteurs caractéristiques de I et F. Alors u_n est le nombre de chemins dans G de longueur n allant d'un sommet de I à un sommet de F.

Exemple 11.4.2. *Soit $(u_n)_{n\geq 0}$ une suite ayant la représentation de niveau 2 suivante :*

$$L = \begin{pmatrix} 1 & 1 \end{pmatrix} \quad M = \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix} \quad C = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Posons
$M^n = \begin{pmatrix} u_{1,1}(n) & u_{1,2}(n) \\ u_{2,1}(n) & u_{2,2}(n) \end{pmatrix}$, *alors* $M^{n+1} = \begin{pmatrix} u_{1,1}(n)+u_{1,2}(n) & 2.u_{1,1}(n) \\ u_{2,1}(n)+u_{2,2}(n) & 2.u_{2,1}(n) \end{pmatrix}$
et $(u_n)_{n\geq 0}$ *est la suite définie par :*

$$u_n = u_{1,1}(n) + u_{1,2}(n)$$

avec :
$u_{1,1}(0) = 1 \;\; u_{1,1}(n+1) = u_{1,1}(n) + u_{1,2}(n) \quad u_{1,2}(0) = 0 \;\; u_{1,2}(n+1) = 2 \cdot u_{1,1}(n)$
$u_{2,1}(0) = 0 \;\; u_{1,1}(n+1) = u_{2,1}(n) + u_{2,2}(n) \quad u_{2,2}(0) = 1 \;\; u_{1,2}(n+1) = 2 \cdot u_{2,1}(n).$

Proposition 11.4.3. *Si* $(u_n)_{n\geq 0}$ *est une suite* $\mathbb{N}$*-rationnelle, alors* $(u_n)_{n\geq 0} \in \mathbb{S}_2$.

Construction : Supposons que u est définie par $L \in \mathbb{B}^{1\times d}$, $M \in \mathbb{N}^{d\times d}$, $C \in \mathbb{B}^{d\times 1}$.

Soit $\mathcal{A} = (Q, \{\alpha\}, (A_1, A_2), \Delta, q_0)$ avec : $Q = \{q_0\}$, $A_1 = \{a_1\}$ $A_2 = \{a_{2,i,j}\}_{1\leq i,j\leq d} \cup \{a_2, b_2\}$, et la relation de transition est définie par :

(1) $\Delta(q_0, \alpha, a_{2,i,i}) = (q_0, \mathrm{pop}_2)$ et si $i \neq j$, $\delta(q_0, \varepsilon, a_{2,i,j}) = (\mathrm{pop}_2, q_0)$,

(2) $\Delta(q_0, \varepsilon, a_{2,i,j}a_1) = (\mathrm{pop}_1\, \mathrm{change}_{b_2} \prod_{1\leq l\leq d} (\mathrm{push}_{a_{2,i,l}})^{m_{l,j}}, q_0)$,

(3) $\Delta(q_0, \varepsilon, a_2a_1) = \Delta(q_0, \varepsilon, a_2) = (\mathrm{change}_{b_2} \prod_{1\leq j\leq d} (\prod_{1\leq i\leq d} (\mathrm{push}_{a_{2,i,j}})^{l_i})^{c_j}, q_0)$,

(4) $\Delta(q_0, \varepsilon, b_2a_1) = \Delta(q_0, \varepsilon, b_2) = (\mathrm{pop}_2, q_0)$.

Preuve : Considérons les suites $u_{i,j}$ définies par la récurrence suivante :

$$u_{i,j}(n+1) = \sum_{1\leq l\leq d} u_{i,l}(n) \cdot m_{l,j},$$

$$u_{i,i}(0) = 1, \;\; u_{i,j}(0) = 0 \text{ pour } i \neq j.$$

Montrons par induction sur $n \geq 0$ la propriété auxiliaire $P(n)$:
$\forall 1 \leq i, j \leq d$,

$$(q_0 a_{2,i,j} [a_1{}^n] q_0) \to^*_{\mathcal{A}} \alpha^{u_{i,j}(n)}. \tag{17}$$

Base : $n = 0$. Les transitions (1) assurent que $P(0)$ est vraie.
Pas d'induction : Supposons $P(n)$ pour $n \geq 0$.

$$\begin{aligned}
(q_0 a_{2,i,j} [a_1{}^{n+1}] q_0) \quad &\to_{\mathcal{A}} \quad (q_0 \prod_{1\leq l\leq d} (a_{2,i,l}[a_1{}^n])^{m_{l,j}} b_2 [a_1{}^n] q_0) \text{ (par transition (2))} \\
&\to^*_{\mathcal{A}} \quad \prod_{1\leq l\leq d} (q_0 a_{2,i,l} [a_1{}^n] q_0)^{m_{l,j}} (q_0, b_2[a_1{}^n], q_0) \;\; \text{(décomposition)} \\
&\to^*_{\mathcal{A}} \quad \prod_{1\leq l\leq d} (q_0 a_{2,i,l} [a_1{}^n] q_0)^{m_{l,j}} \;\; \text{(par transition (4))}.
\end{aligned}$$

Par hypothèse d'induction :

$$(q_0 a_{2,i,l} [a_1{}^n] q_0) \to^*_{\mathcal{A}} \alpha^{u_{i,l}(n)}.$$

En composant les dérivations précédentes, nous obtenons :

$$(q_0 a_{2,i,j}[{a_1}^{n+1}]q_0) \to^*_{\mathcal{A}} \prod_{1\le l\le d} (\alpha^{u_{i,l}(n)})^{m_{l,j}} = \alpha^{u_{i,j}(n+1)}.$$

Ceci prouve $P(n+1)$ et la propriété (17) est donc établie pour tout n.
Examinons maintenant la suite u. Pour tout $n \ge 0$:

$$u(n) = \sum_{1\le i\le d} \sum_{1\le j\le d} \ell_i \cdot u_{i,j}(n) \cdot c_j. \tag{18}$$

En appliquant la transition (3) suivie par des règles de décompositions, et de la transition (4), nous obtenons :

$$\begin{aligned}(q_0 a_2[{a_1}^n]q_0) &\to^*_{\mathcal{A}} (q_0 \prod_{1\le j\le d} (\prod_{1\le i\le d} (a_{2,i,j}[{a_1}^n])^{l_i})^{c_j} \cdot b_2[{a_1}^n]q_0) \\ &\to^*_{\mathcal{A}} \prod_{1\le j\le d} (\prod_{1\le i\le d} (q_0 a_{2,i,j}[{a_1}^n]q_0)^{l_i\cdot c_j}) \end{aligned} \tag{19}$$

et d'après $P(n)$ nous déduisons que

$$\prod_{1\le i\le d} (q_0 a_{2,i,j}[{a_1}^n]q_0)^{l_i\cdot c_j} \to^*_{\mathcal{A}} \prod_{1\le i\le d} \alpha^{u_{i,j}(n)\cdot l_i\cdot c_j}. \tag{20}$$

En combinant les dérivations (19) et (20) nous obtenons, par la formule (18) :

$$(q_0 a_2[{a_1}^n]q_0) \to^*_{\mathcal{A}} \alpha^{u(n)}.$$

□

Lemme 11.4.4. *Soit $(u_n)_{n\ge 0}$ la suite définie par $u_{n+1} = (u_n)^d$, $d \ge 2$ et $u_0 = c \in \mathbb{N}$. Alors $(u_n)_{n\ge 0} \in \mathbb{S}_3$.*

Construction : Posons $\mathcal{A} = (\{q_0\}, \{\alpha\}, (\{a_1\}, \{a_2\}, \{a_3\}), \Delta, q_0, \bot)$ avec :

(1) $\Delta(q_0, \varepsilon, a_3a_2a_1) = (\mathrm{pop}_1(\mathrm{push}_{a_2})^{d-1}, q_0)$,

(2) $\Delta(q_0, \varepsilon, a_3a_2) = (\mathrm{pop}_2(\mathrm{push}_{a_3})^{c-1}, q_0)$ si $c \ge 1$ et $= (\mathrm{pop}_3, q_0)$ sinon,

(3) $\Delta(q_0, \alpha, a_3) = (\mathrm{pop}_3, q_0)$.

Preuve : Soit Ω_2 une indéterminée de niveau 2 pour l'automate $\mathcal{A}$ (c'est à dire, $\Omega_2 \notin \{a_1, a_2, a_3\}$). Nous prouvons par induction sur $n \ge 0$, la propriété $\mathbf{P}(n)$ suivante :

$$(q_0 a_3[a_2[{a_1}^n]\Omega_2]q_0) \to^*_{\mathcal{A}} (q_0 a_3[\Omega_2]q_0)^{u_n}.$$

Base : $n = 0$
Par la transition (2) et les règles de décomposition :

$$(q_0 a_3[a_2[\varepsilon]\Omega_2]q_0) \to_{\mathcal{A}} (q_0 (a_3[\Omega_2])^c q_0) \to^*_{\mathcal{A}} (q_0 a_3[\Omega]q_0)^c. \tag{21}$$

La propriété $\mathbf{P}(0)$ est donc prouvée.
Pas d'induction : Supposons $\mathbf{P}(n)$ pour $n \geq 0$.
En utilisant la transition (1), nous avons :

$$(q_0 a_3[a_2[{a_1}^{n+1}]\Omega_2]q_0) \to_{\mathcal{A}} (q_0 a_3[(a_2[{a_1}^n])^d \Omega_2]q_0). \tag{22}$$

Soit $i \geq 1$ et $T_i = (a_2[{a_1}^n])^{i-1}\Omega$. En substituant le 2-terme T_i à l'indéterminée Ω dans $\mathbf{P}(n)$ nous obtenons :

$$(q_0 a_3[(a_2[{a_1}^n])^i \Omega_2]q_0) \to^*_{\mathcal{A}} (q_0 a_3[(a_2[{a_1}^n])^{i-1}\Omega_2]q_0)^{u_n}.$$

En composant ces dérivations (pour $1 \leq i \leq d$), nous obtenons :

$$(q_0 a_3[(a_2[{a_1}^n])^d \Omega_2]q_0) \to^*_{\mathcal{A}} (q_0 a_3[\Omega_2]q_0)^{(u_n)^d}. \tag{23}$$

En composant maintenant les dérivations (22) et (23) :

$$(q_0 a_3[a_2[{a_1}^{n+1}]\Omega]q_0) \to^*_{\mathcal{A}} (q_0 a_3[\Omega]q_0)^{(u_n)^d} = (q_0 a_3[\Omega]q_0)^{u_{n+1}}$$

i.e., $\mathbf{P}(n+1)$. La propriété $\mathbf{P}(n)$ est donc vérifiée pour tout $n \geq 0$.

La transition (3) exprimée comme une dérivation donne :

$$(q_0 a_3[\varepsilon]q_0) \to_{\mathcal{A}} \alpha.$$

Nous pouvons alors conclure que, pour tout $n \in \mathbb{N}$

$$(q_0 a_3[a_2[{a_1}^n]]q_0) \to^*_{\mathcal{A}} \alpha^{u_n}.$$

□

Proposition 11.4.5. *Soit $P(X) = \sum_{0 \leq i \leq d} \gamma_i X^i$ un polynôme avec coefficients $\gamma_i \in \mathbb{N}$ et $(u_n)_{n \geq 0}$ la suite définie par $u_{n+1} = P(u_n)$ et $u_0 = \gamma \in \mathbb{N}^+$.*

Alors $(u_n)_{n \geq 0} \in \mathbb{S}_3$.

Nous utilisons ici la même idée que la preuve précédente.
Construction : Posons $\mathcal{A} = (\{q_0, q\}, \{\alpha\}, (\{a_1\}, \{a_2\}, \{a_3, b_3, c_{3,0}, \ldots c_{3,d}\}), \Delta, q_0, \bot)$
avec :

(1) $\Delta(q_0, \varepsilon, a_3 a_2 a_1) = (\mathrm{pop}_1 \,\mathrm{change}_{c_{3,1}}(\mathrm{push}_{c_{3,0}})^{\gamma_0}, q_0)$,

(2) $\Delta(q_0, \varepsilon, c_{3,0} a_2 a_1) = \Delta(q_0, \varepsilon, c_{3,0} a_2) = (\mathrm{pop}_2 \,\mathrm{change}_{a_3}, q_0)$

(3) $\Delta(q_0, \varepsilon, a_3 a_2 a_1) = \Delta(q_0, \varepsilon, a_3 a_2) = (\mathrm{change}_{c_{3,2}}(\mathrm{push}_{a_3})^{\gamma_1}, q_0)$,

(4) pour tout $2 \leq i \leq d$, $\Delta(q_0, \varepsilon, c_{3,i} a_2 a_1) = \Delta(q_0, \varepsilon, c_{3,i} a_2) = (\mathrm{push}_{a_2}, q)$,

(5) pour tout $2 \leq i \leq d$, $\Delta(q, \varepsilon, c_{3,i} a_2 a_1) = \Delta(q, \varepsilon, c_{3,i} a_2) = (\mathrm{change}_{c_{3,i+1}}(\mathrm{push}_{a_3})^{\gamma_i}, q_0)$,

(6) $\Delta(q, \varepsilon, c_{3,d} a_2 a_1) = \Delta(q, \varepsilon, c_{3,d} a_2) = (\mathrm{change}_{a_3}(\mathrm{push}_{a_3})^{\gamma_d - 1}, q_0)$,

(7.1) $\Delta(q_0, \varepsilon, a_3 a_2) = (\mathrm{change}_3(b_3)\,\mathrm{push}_3(b_3)^{\gamma - 1}, q_0)$,

(7.2) $\Delta(q_0, \varepsilon, b_3 a_2) = (\mathrm{change}_3(a_3)\,\mathrm{pop}_2, q_0)$,

(8) $\Delta(q_0, \alpha, a_3) = (\mathrm{pop}_3, q_0)$.

Preuve : Soit Ω_2 une indéterminée de niveau 2. montrons que, pour tout $n \geq 0$, la propriété $\mathbf{P}(n)$ suivante est vraie :

$$(q_0 a_3[a_2[{a_1}^n]\Omega_2]q_0) \rightarrow_{\mathcal{A}}^* (q_0\Omega_2 q_0)^{u(n)}.$$

Commençons par vérifier que pour tout $n \geq 0$,

$$(q_0 a_3[a_2[{a_1}^{n+1}]\Omega_2]q_0) \rightarrow_{\mathcal{A}}^* \prod_{i=0}^{d} (q_0 a_3[(a_2[{a_1}^n])^i\Omega_2]q_0)^{\gamma_i}. \tag{24}$$

Une telle dérivation peut être décomposée comme suit :

$$\begin{array}{ll}
 & (q_0 a_3[a_2[{a_1}^{n+1}]\Omega_2]q_0) \\
\rightarrow_{\mathcal{A}} & (q_0(\mathrm{C}_0[a_2[{a_1}^n]\Omega_2])^{\gamma_0}\mathrm{C}_{3,1}[a_2[{a_1}^n]\Omega_2]q_0) \text{ (par transition (1))} \\
\rightarrow_{\mathcal{A}}^* & (q_0\mathrm{C}_{3,0}[a_2[{a_1}^n]\Omega_2]q_0)^{\gamma_0}(q_0\mathrm{C}_{3,1}[a_2[{a_1}^n]\Omega_2]q_0) \text{ (par règle de décomposition)} \\
\rightarrow_{\mathcal{A}} & (q_0 a_3[\Omega_2]q_0)^{\gamma_0}(q_0\mathrm{C}_{3,1}[a_2[{a_1}^n]\Omega_2]q_0) \text{ (par transition (2))} \\
\rightarrow_{\mathcal{A}}^* & (q_0 a_3[\Omega_2]q_0)^{\gamma_0}(q_0(a_3[a_2[{a_1}^n]\Omega_2])^{\gamma_1}\mathrm{C}_{3,2}[a_2[{a_1}^n]\Omega_2]q_0) \text{ (par transition (3))} \\
\rightarrow_{\mathcal{A}}^* & (q_0 a_3[\Omega_2]q_0)^{\gamma_0}(q_0(a_3[a_2[{a_1}^n]\Omega_2]q_0)^{\gamma_1}(q_0\mathrm{C}_{3,2}[a_2[{a_1}^n]\Omega_2]q_0) \text{ (par décomposition)} \\
\rightarrow_{\mathcal{A}}^* & (q_0 a_3[\Omega_2]q_0)^{\gamma_0}\prod_{i=1}^{d}(q_0(a_3[(a_2[{a_1}^n])^i\Omega_2]q_0)^{\gamma_i} \text{ (en appliquant } d-2 \text{ fois } (4,5) \text{ puis } (4,6)).
\end{array}$$

Prouvons maintenant $\mathbf{P}(n)$ par induction :
Base : $n = 0$
La dérivation suivante est valide :

$$\begin{array}{rcll}
(q_0 a_3[a_2[\varepsilon]\Omega_2]q_0) \rightarrow_{\mathcal{A}} & (q_0(b_3[a_2[\varepsilon]\Omega_2])^{\gamma}q_0) & \text{(par (7.1))} \\
\rightarrow_{\mathcal{A}}^* & (q_0 b_3[a_2[\varepsilon]\Omega_2]q_0)^{\gamma} & \text{(par règles de décomposition)} \\
\rightarrow_{\mathcal{A}}^* & (q_0 a_3[\Omega_2]q_0)^{\gamma} & \text{(par (7.2))}.
\end{array}$$

Pas d'induction : Supposons $\mathbf{P}(n)$.
Dans ce cas, on peut montrer en utilisant la substitution $\Omega_2 \leftarrow (a[{a_1}^n])^{i-1}\Omega_2$, que :

$$(q_0 a_3[(a_2[{a_1}^n])^i\Omega_2]q_0) \rightarrow_{\mathcal{A}}^* (q_0 a_3[\Omega_2]q_0)^{u(n)^i}.$$

Ces dérivations permettent d'obtenir :

$$\prod_{i=0}^{d}(q_0 a_3[(a_2[{a_1}^n])^i\Omega_2]q_0)^{\gamma_i} \rightarrow_{\mathcal{A}}^* (q_0 a_3[\Omega_2]q_0)^{P(u(n))} = (q_0 a_3[\Omega]q_0)^{u(n+1)}.$$

La combinaison des dérivations (24) et de la dérivation précédente prouve $\mathbf{P}(n+1)$. La propriété $\mathbf{P}(n)$ est donc vérifiée pour tout $n \geq 0$.
En substituant ε à Ω_2 dans $\mathbf{P}(n)$ et en appliquant la transition (8), nous concluons que

$$(q_0 a_3[a_2[{a_1}^n]]q_0) \rightarrow_{\mathcal{A}}^* \alpha^{u(n)}.$$

□

Proposition 11.4.6. *Soient $P_i(X_1, \ldots, X_p)$, $(1 \leq i \leq p)$ des polynômes à coefficients dans $\mathbb{N}$, $c_1, \ldots, c_i, \ldots c_p \in \mathbb{N}$ et, u_i, $(1 \leq i \leq p)$ la suite définie par $u_i(n+1) = P_i(u_1(n), \ldots, u_p(n))$, et $u_i(0) = c_i$. Alors $u_1 \in \mathbb{S}_3$.*

Esquisse de preuve : Supposons que pour tout $1 \leq i \leq p$,

$$P_i(X_1, \ldots X_p) = \sum_{j=0}^{\nu_i} c_{i,j} X_1^{d_{i,j,1}} \cdots X_p^{d_{i,j,p}}.$$

En suivant les mêmes lignes que dans la preuve de la proposition 11.4.5, il est possible de construire un 3-ACD satisfaisant pour tout $1 \leq i \leq p$,

$$(q_0 a_3[\mathrm{P}_{2,i}[{a_1}^{n+1}]\Omega_2]q_0) \rightarrow^* \prod_{j=0}^{\nu_i} (q_0 a_3[(\prod_{\ell=1}^{p} (\mathrm{P}_{2,\ell}[{a_1}^n])^{d_{i,j,\ell}}) \cdot \Omega_2]q_0)^{c_{i,j}}$$

ainsi que

$$(q_0 a_3[\mathrm{P}_{2,i}[\varepsilon]\Omega_2]q_0) \rightarrow^* (q_0 a_2[\Omega_2]q_0)^{c_i}.$$

Ces dérivations impliquent, par induction sur n, que :

$$(q_0, a_3[\mathrm{P}_{2,i}[{a_1}^n]]q_0) \rightarrow^* \alpha^{u_i(n)}.$$

□

Proposition 11.4.7. *Soit s une suite entière strictement croissante et telle que $s(0) = 0$, alors $s^{-1} \in \mathbb{S}_2^{s(\mathbb{N})}$.*

Preuve : $\mathcal{A} = (\{q_0\}, \{\alpha\}, (\{a_1\}, \{a_2\}), s(\mathbb{N}), \Delta, q_0)$ avec
$\Delta(q_0, \varepsilon, a_2, o) = (q_0, \mathrm{pop}_2)$ pour $o \in \{0, 1\}$,
$\Delta(q_0, \varepsilon, a_2 a_1, 0) = \Delta(q_0, \alpha, a_2 a_1, 1) = (\mathrm{pop}_1, q_0)$.

Partant d'une configuration $(q_0, \sigma, a_2[{a_1}^n])$, l'automate dépile itérativement le niveau 1, en lisant à chaque itération la lettre α en entrée ssi l'entier codé au niveau 1 est dans $s(\mathbb{N})$. Finalement, lorsque l'automate a vidé le niveau 1, il a lu sur la bande d'entrée exactement autant de α qu'il y a d'éléments dans $[1, n] \cap s(\mathbb{N})$, c.a.d., $s^{-1}(n)$.

□

11.5 Opérations sur suites/automates.

Nous étudions dans cette section les propriétés de clôture de la classe $\mathbb{S}_k$. L'union de tous les $\mathbb{S}_k$ est clos par les opérations classiques telles que la somme, le produit, la composition, le produit de convolution, ainsi que par des opérations plus complexes comme la résolution de systèmes d'équation récurrentes polynomiales à coefficients dans $\mathbb{S}_k$. Un soin particulier est apporté à la conservation du niveau des suites, par exemple $\mathbb{S}_k^{\vec{N}}$ est stable par somme à partir de $k = 2$ et par produit à partir de $k = 3$. Nous concluons la section en énumérant dans le théorème 11.5.19 toutes les propriétés établies.

Nous débutons par deux lemmes techniques qui seront utilisés dans toutes les constructions à venir. Il s'agit en fait de deux versions du même lemme : une version faible et une version

forte, qui permettent, à partir d'un k-ACD$^{\vec{N}}$ calculant une suite s, de construire un nouveau k-ACD$^{\vec{N}}$ sachant faire la copie $s(n)$ fois d'une configuration particulière. Nous construisons cet automate de façon à ce qu'il soit prêt à être composé avec un autre.

Nous utiliserons dans toute cette section la notation suivante : pour tout $k \geq 2$, $i \in [1, k-1]$,

$$T_{k,i}[\Omega_{i-1}] := a_k[a_{k-1}[\cdots[a_i[\Omega_{i-1}]]\cdots]]$$

En particulier $T_{k,k}[\Omega_{k-1}] = a_k[\Omega_{k-1}]$ et $T_{k,k-1}[\Omega_k] = \Omega_k$.

Lemme 11.5.1 (Forme normale faible). *Soit $s \in \mathbb{S}^{\vec{N}}_{k+1}$, $k \geq 1$ et $\mathcal{A} \in (k+1)$-ACD$^{\vec{N}}$ défini sur les alphabets de pile $A_1, \ldots, A_k$ où $a_1 \in A_1, \ldots, a_{k+1} \in A_{k+1}$ et vérifiant,*
(**H1**) $\forall n \geq 0$, $(q_0, \alpha^{s(n)}, a_{k+1}[a_k[\ldots[a_2[a_1{}^n]]\ldots]]) \vdash^*_{\mathcal{A}} (q_0, \varepsilon, \varepsilon)$.
(**H2**) *$\mathcal{A}$ ne contient pas de membre gauche de la forme (q, α, ε) ou $(q, \varepsilon, \varepsilon)$.*

Alors on peut construire $\mathcal{A}' \in k$-ACD$^{\vec{N}}$ défini sur les alphabets de piles $(A_1 \cup A'_1, \ldots, A_k \cup A_k{}')$, où A'_k contient un symbole particulier A_k, dont l'ensemble d'états contient q_0 et tel que :
(**P1**) $(q_0, a_{k+1}[a_k[\ldots[a_2[a_1{}^n]]\ldots]], q_0) \rightarrow^*_{\mathcal{B}} (q_0, \mathrm{A}_{k+1}[\varepsilon], q_0)^{s(n)}$.
(**P2**) *Δ' ne contient pas de membre gauche de la forme $(q_0, \varepsilon, \varepsilon)$.*
(**P3**) *Δ' ne contient pas de membre gauche de la forme $(q_0, \varepsilon, \mathrm{A}_{k+1} \cdot w)$.*

Construction : Supposons que $\mathcal{A}$ est un automate vérifiant les hypothèses (**H1**), (**H2**) et donné par $\mathcal{A} = (Q, \{\alpha\}, (A_1, \ldots, A_{k+1}), \vec{N}, \Delta, q_0)$. Posons $B_{k+1} = A_{k+1} \cup \{\mathrm{A}_{k+1}, \mathrm{B}_{k+1}\} \cup \{(b_{k+1}, \delta) \mid b_{k+1} \in A_{k+1},\ \delta \in \Delta\}$ et

$$\mathcal{B} = (Q, \emptyset, (A_1, \ldots, A_k, B_{k+1}), \vec{N}, \Delta', q_0)$$

où Δ' est composé des transitions suivantes :

- pour tout $\Delta(p, \varepsilon, w, \vec{o}) = (\mathrm{instr}, q)$
  (1) $\Delta'(p, \varepsilon, w, \vec{o}) = (\mathrm{instr}, q)$
- pour tout $b_{k+1} \in A_{k+1}$ et $\delta = (p, \alpha, b_k w, \vec{o}, \mathrm{instr}, q) \in \Delta$
  (2.1) $\Delta'(p, \varepsilon, b_{k+1} w, \vec{o}) = (\mathrm{change}_{(b_{k+1}, \delta)}\, \mathrm{push}_{\mathrm{B}_{k+1}}, q_0)$
  (2.2) $\Delta'(q_0, \varepsilon, (b_{k+1}, \delta) w, \vec{o}) = (\mathrm{change}_{b_{k+1}}\, \mathrm{instr}, q)$
- pour tous $w \neq \varepsilon \in \mathrm{top}(k\text{-}\mathsf{NPile}(A_1, \ldots, A_k))$, $\vec{o} \in \{0,1\}^{|\vec{N}|}$,
  (3.1) $\Delta'(q_0, \varepsilon, \mathrm{B}_{k+1} w, \vec{o}) = (\mathrm{pop}_k, q_0)$
- pour tout $\vec{o} \in \{0,1\}^{|\vec{N}|}$,
  (3.2) $\Delta'(q_0, \varepsilon, \mathrm{B}_{k+1}, \vec{o}) = (\mathrm{push}_{\mathrm{A}_{k+1}}, q_0)$

Preuve : Prouvons maintenant la validité de la construction.
Déterminisme et conditions (**P2,P3**) : Vérifions que $\mathcal{B}$ est déterministe. L'automate $\mathcal{A}$ étant déterministe, deux transitions distinctes de types 1 ou 2 sont donc toujours incompatibles. Les transitions de type 3 sont incompatibles entre elles et puisque B_{k+1} est un nouveau symbole, chacune d'elles est incompatible avec toute transition de type 1 ou 2. L'automate $\mathcal{B}$ est donc déterministe.

L'automate $\mathcal{A}$ vérifiant l'hypothèse (**H2**), il est évident que $\mathcal{B}$ vérifie (**P2**).

Enfin, la condition (**P3**) est vérifiée par les transitions issues de $\mathcal{A}$ (de type 1 et 2) car A_{k+1} est un nouveau symbole, et comme nous n'avons pas posé de transitions utilisant ce symbole, donc la condition (**P3**) est vérifiée par $\mathcal{B}$.

Condition (**P1**) : Pour prouver que $\mathcal{B}$ vérifie la condition (**P1**), nous établissons les deux implications suivantes :

pour tous $p, q \in Q$, $\omega, \omega' \in k+1\text{-Pile}(A_1, \ldots, A_k, B_{k+1})$

$$(p, \varepsilon, \omega) \vdash_{\mathcal{A}}^{*} (q, \varepsilon, \omega') \implies (p\omega q_0) \rightarrow_{\mathcal{B}}^{*} (q\omega' q_0) \quad (25)$$

$$(p, \alpha, \omega) \vdash_{\mathcal{A}} (q, \varepsilon, \omega') \implies (p\omega q_0) \rightarrow_{\mathcal{B}}^{*} (q_0 \mathrm{A}_{k+1}[\varepsilon] q_0)(q\omega' q_0) \quad (26)$$

Notons que nous laissons ouverte la possibilité que ω, ω' contiennent des occurrences des lettres n'appartenant pas à A_{k+1}. La relation $\vdash_{\mathcal{A}}^{*}$ est définie à partir des transitions de $\mathcal{A}$, mais appliquée aux états totaux dans $Q \times k\text{-Pile}(A_1, \ldots, A_k, B_{k+1})$.
La propriété (25) est obtenue par simple traduction, en terme de dérivation, des transitions de type (1). Prouvons l'implication (26) :
supposons $\omega = b_{k+1}[\omega_1]\omega''$, $\omega' = \mathrm{instr}(\omega)$ et $(p, \alpha, \omega) \vdash_{\delta} (q, \varepsilon, \omega')$, pour $\delta \in \Delta$. La dérivation suivante est valide :

$$\begin{array}{rll}
(p\omega q_0) & \rightarrow_{\mathcal{B}} & (q_0 \mathrm{B}_{k+1}[\omega_1](b_{k+1}, \delta)[\omega_1]\omega'' q_0) \text{ (par (2.1))} \\
& \rightarrow_{\mathcal{B}}^{*} & (q_0 \mathrm{B}_{k+1}[\varepsilon](a, \delta)[\omega_1]\omega'' q_0) \text{ (par itération de (3.1))} \\
& \rightarrow_{\mathcal{B}} & (q_0 \mathrm{A}_{k+1}[\varepsilon](a, \delta)[\omega_1]\omega'' q_0) \text{ (par (3.2))} \\
& \rightarrow_{\mathcal{B}} & (q_0 \mathrm{A}_{k+1}[\varepsilon] q_0)(q_0(a, \delta)[\omega_1]\omega'' q_0) \text{ (par décomposition)} \\
& \rightarrow_{\mathcal{B}} & (q_0 \mathrm{A}_{k+1}[\varepsilon] q_0)(q\omega' q_0) \text{(par (2.2))}
\end{array}$$

En utilisant les implications (25) et (26), et l'hypothèse (**H1**) sur $\mathcal{A}$, une induction évidente sur la longueur de la dérivation décrite dans (**H1**) prouve que pour tout $n \geq 0$,

$$(q_0 a_{k+1}[\cdots[{a_1}^n]\cdots] q_0) \rightarrow_{\mathcal{B}}^{*} (q_0 \mathrm{A}_{k+1}[\varepsilon] q_0)^{s(n)}$$

□

Lemme 11.5.2 (Forme normale forte). *Soit $s \in \mathbb{S}_{k+1}^{\vec{N}}$, $k \geq 2$ et $\mathcal{A} \in (k+1)\text{-ACD}^{\vec{N}}$ défini sur les alphabets de pile $A_1, \ldots, A_k$ où $a_1 \in A_1, \ldots, a_{k+1} \in A_{k+1}$ et vérifiant $\forall n \geq 0$,*
(**H1**) $(q_0, \alpha^{s(n)}, a_{k+1}[a_k[\ldots[a_2[{a_1}^n]]\ldots]]) \vdash_{\mathcal{A}}^{*} (q_0, \varepsilon, \varepsilon)$.
(**H2**) *$\mathcal{A}$ ne contient pas de membre gauche de la forme $(q_0, \alpha, \varepsilon)$ ou $(q_0, \varepsilon, \varepsilon)$.*

Alors on peut construire $\mathcal{B} \in k\text{-ACD}^{\vec{N}}$ défini sur les alphabets de piles $(A_1 \cup A_1', \ldots, A_k \cup {A_k}')$, où A_k' contient un symbole particulier A_k, dont l'ensemble d'états contient q_0 et tel que :
(**Q1**) $\forall \Omega_k \in \mathfrak{I}_k$, $(q_0, a_{k+1}[a_k[\ldots[a_2[{a_1}^n]]\ldots]\Omega_k], q_0) \rightarrow_{\mathcal{B}}^{*} (q_0, \mathrm{A}_{k+1}[\Omega_k], q_0)^{s(n)}$
(**Q2**) *Δ' ne contient pas de membre gauche de la forme $(q_0, \varepsilon, \varepsilon)$.*
(**Q3**) *Δ' ne contient pas de membre gauche de la forme $(q_0, \varepsilon, \mathrm{A}_{k+1} \cdot w)$.*

Preuve : Considérons les dérivations suivantes :
Règle d'initialisation (D0) :

$$(q_0 a_{k+1}[T_{k,2}[{a_1}^n]\Omega_k] q_0) \rightarrow_{\mathcal{B}}^{*} (q_0 b_{k+1}[T_{k,2}[{a_1}^n]\mathrm{B}_k[T_{k-1,2}[{a_1}^n]]\Omega_k] q_0)$$

Règle de calcul de f (D1) :

$$(q_0 b_{k+1}[T_{k,2}[{a_1}^n]\mathrm{B}_k[T_{k-1,2}[{a_1}^n]]\Omega_k] q_0) \rightarrow_{\mathcal{B}}^{*} (q_0 \mathrm{A}_{k+1}[\mathrm{B}_k[T_{k-1,2}[{a_1}^n]]\Omega_k] q_0)^{f(n)}$$

Règle de terminaison (D2) :

$$(q_0 \mathrm{A}_{k+1}[b_k[T_{k-1,2}[{a_1}^n]]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}'} (q_0 \mathrm{A}_{k+1}[\Omega_k]q_0)$$

Tout automate réalisant ces trois dérivations réalise également la dérivation (**Q1**) :

$$\begin{array}{rcl}(q_0, a_{k+1}[T_{k,2}[{a_1}^n]\Omega_k], q_0) & \rightarrow^*_{\mathcal{B}} & (q_0 b_{k+1}[T_{k,2}[{a_1}^n]\mathrm{B}_k[T_{k-1,2}[{a_1}^n]]\Omega_k]q_0) \\ & \rightarrow^*_{\mathcal{B}} & (q_0 \mathrm{A}_{k+1}[b_k[T_{k-1,2}[{a_1}^n]]\Omega_k]q_0)^{f(n)} \\ & \rightarrow^*_{\mathcal{B}} & (q_0 \mathrm{A}_{k+1}[\Omega_k]q_0)^{f(n)}\end{array}$$

Pour prouver le lemme, il nous suffit donc de construire un automate $\mathcal{B}$ déterministe pour lequel les dérivations (D0), (D1) et (D2) sont valides et vérifiant les conditions (**Q2,Q3**).
Construction : En utilisant le lemme 11.5.1, et par un renommage approprié des lettres des alphabets de piles, nous obtenons un automate $\mathcal{A} = (Q, \emptyset, (A_1, \ldots, A_{k+1}), \vec{N}, \Delta', q_0)$ déterministe et vérifiant les conditions (**P1**$(b_{k+1}a_k \cdots a_1, \mathrm{A}_{k+1})$), (**P2**) et (**P3**$(\mathrm{A}_{k+1})$).

Posons $\mathcal{B} = (Q, \emptyset, (B_1, \ldots, B_{k+1}), \vec{N}, \Delta \cup \Delta', q_0)$, avec $B_{k+1} = A_{k+1} \cup \{a_{k+1}\}$, $B_k = A_k \cup \{\mathrm{B}_k\}$, $B_i = A_i$ pour $1 \leq i \leq k-1$, et Δ est composé des transitions suivantes :

- pour les symboles précis $a_1, a_2, \ldots a_k, b_{k+1}$ utilisés dans (**P1**), pour tout $\vec{o} \in \{0,1\}^{|\vec{N}|}$
  (0) $\delta(q_0, \varepsilon, a_{k+1} \cdots a_2 a_1, \vec{o}) = (\mathrm{change}_{b_{k+1}} \mathrm{change}_{\mathrm{B}_k} \mathrm{push}_{a_k}, q_0)$
- pour tout $(q, \varepsilon, c_{k+1}, \chi_{\vec{N}}(0), \mathrm{instr}, p) \in \Delta'$, $c_{k+1} \in A_{k+1}$ quelconque, pour tout $\vec{o} \in \{0,1\}^{|\vec{N}|}$
  (1) $\Delta(q, \varepsilon, c_{k+1}\mathrm{B}_k a_{k-1} \cdots a_2 a_1, \vec{o}) = \Delta(q, \varepsilon, c_{k+1}\mathrm{B}_k a_{k-1} \cdots a_2, \vec{o}) = (\mathrm{instr}, p)$
- pour tous $w \in \mathrm{top}((k-1)\text{-}\mathsf{NPile}(A_1, \ldots, A_{k-1}))$, $\vec{o} \in \{0,1\}^{|\vec{N}|}$
  (2) $\Delta'(q_0, \varepsilon, \mathrm{A}_{k+1}\mathrm{B}_k w, \vec{o}) = (\mathrm{pop}_k, q_0)$.

Déterminisme et conditions (**Q2, Q3**) : L'automate $\mathcal{A}$ est déterministe et puisque a_{k+1}, B_k sont des nouveaux symboles, l'ajout des transitions (0) et (2) n'introduisent pas de non déterministe. De même, les transitions de types (1) sont incompatibles avec toute transition de Δ' où d'un autre type, et puisque $\mathcal{A}$ est déterministe, pour tout couple $(q, c_{k+1}) \in Q \times A_{k+1}$, il existe une unique transition ayant pour membre gauche $(q, \varepsilon, c_{k+1}, \chi_{\vec{N}}(0))$ et les transitions de type (1) sont donc toutes incompatibles entre elles. L'automate $\mathcal{B}$ est donc déterministe. De plus, $\mathcal{A}$ vérifie (**P2**) et (**P3**(A_{k+1})), et l'ajout des transitions (0),(1), et (2), conserve ces propriétés. L'automate $\mathcal{B}$ vérifie donc (**Q2**) et (**Q3**(A_{k+1})).
Condition (**Q1**) : D'après la discussion précédant la construction, il nous suffit de montrer que les dérivation (D0), (D1) et (D2) sont réalisées par l'automate. La dérivation (D0) est obtenue par application de la transition 0, et (D2) est réalisée par une transition de type (2). Il nous reste donc à vérifier que (D1) est une dérivation valide.

Définissons pour tout $n \geq 0$, l'application

$$\tau_n : (k+1)\text{-}\mathsf{NPile}(A_1, \ldots, A_{k+1}) \rightarrow k\text{-}\mathsf{Terme}(B_1, \ldots, B_{k+1})$$

associant à toute pile le terme obtenu en ajoutant $\mathrm{B}_k[T_{k-1,2}[{a_1}^n]]\Omega_k$ au fond de chaque pile de niveau k :

- $\forall \omega = c_{k+1}[\omega_1]\omega' \in (k+1)\text{-}\mathsf{Pile}$, $\tau_n(\omega) = c_{k+1}[\omega_1 \mathrm{B}_k[T_{k-1,2}[{a_1}^n]]\Omega_k]\tau_n(\omega)$
- $\tau_n(\varepsilon) = \varepsilon$.

Pour tout $\omega, \omega' \in (k+1)\text{-}\mathsf{NPile}$, $p, q \in Q$, $n \geq 0$

$$(p, \varepsilon, \omega) \vdash_{\mathcal{A}} (q, \varepsilon, \omega') \implies (p, \varepsilon, \tau_n(\omega)) \vdash_{\mathcal{B}} (q, \varepsilon, \tau_n(\omega')) \tag{27}$$

Cette propriété se vérifie facilement :

– si $\mathrm{top}_k(\omega) \neq \varepsilon$, alors $\mathrm{top}_k(\omega) = \mathrm{top}_k(\tau_n(\omega))$ et la transition appliquée au membre gauche de (27) est également applicable à $(p, \varepsilon, \tau_n(\omega))$ donc

$$(p, \varepsilon, \tau_n(\omega)) \vdash_{\mathcal{B}} (q, \varepsilon, \tau_n(\omega'))$$

– sinon, $\omega = c_{k+1}[\varepsilon]\omega'$, alors l'instruction appliquée au membre gauche de 27 est forcément de la forme $(p, \varepsilon, c_{k+1}, \chi_{\vec{N}}(0), \mathrm{instr}, q)$ où instr est soit une $(k+1)$-instruction, soit une instruction push de niveau k.
Alors, il existe $\vec{o} = \chi_{\vec{N}}(n)$, tel que la transition de type (2) $(p, \varepsilon, b_{k+1}\mathrm{B}_k \cdots a_1, \vec{o}, \mathrm{instr}, q)$ appartient à Δ et

$$(p, \varepsilon, \tau_n(\omega)) \vdash_{\mathcal{B}} (q, \varepsilon, \tau_n(\omega'))$$

Exprimons ce résultat sous forme de dérivations :
pour tous $\omega, \omega_i \in (k+1)$-NPile, $i \in [1, \ell]$, $p, q, q_i, p_i \in Q$, $n, m \geq 0$:

$$(p, \omega, q) \rightarrow_{\mathcal{A}}^{m} \prod_{i=1}^{\ell} (p_i, \omega_i, q_i) \implies (p, \tau_n(\omega), q) \rightarrow_{\mathcal{B}}^{m} \prod_{i=1}^{\ell} (p_i, \tau_n(\omega_i), q_i) \tag{28}$$

Cette implication se vérifie facilement par une induction sur $m \geq 0$. Si la règle appliquée est une règle de décomposition, elle s'applique également à $(p, \tau_n(\omega), q)$ et la propriété est donc vérifiée. Si la règle appliquée est issue d'une transition, alors (27) implique immédiatement (28).

Nous pouvons maintenant achever la preuve du lemme en prouvant que $\mathcal{B}$ réalise la dérivation (D1). En substituant la dérivation $(\mathbf{P1}(b_{k+1}a_k \cdots a_1, \mathrm{A}_{k+1}))$ au membre gauche de l'implication (28), nous obtenons directement

$$(q_0, \tau_n(b_{k+1}[T_{k,2}[{a_1}^n]]), q_0) \rightarrow_{\mathcal{A}'}^{*} (q_0, \tau_n(\mathrm{A}_{k+1}[\varepsilon])q_0)$$

c'est à dire (D1).

□

Remarque 11.5.3.

1. *Ajoutons aux transitions de l'automate $\mathcal{B}$ construit dans le lemme 11.5.1 (resp. 11.5.2), les transitions $(q_0, \alpha, \mathrm{A}_{k+1}, \vec{o}, \mathrm{pop}_1, q_0)$ (pour $\vec{o}$ quelconque), nous obtenons ainsi un automate $\mathcal{B}'$ vérifiant les hypothèses* (**H0**), (**H1**) *de la définition d'automate calculant une suite (définition 11.2.1).*
2. *Les propriétés* (**P1**) *(resp.* (**Q1**)*) rendent l'automate $\mathcal{B}$ prêt à être combiné avec un autre automate : il suffit d'ajouter des transitions partant de $q_0\mathrm{A}_k[\omega]$ pour ω bien choisi, et amenant à une configuration d'un autre automate déterministe. Les propriétés* (**P2**, **P3**) *(resp.* (**Q2**, **Q3**)*) permettent que le nouvel automate ainsi composé reste déterministe.*
3. *La version forte du lemme n'est valable que pour $k \geq 2$. Le cas $k = 1$ est particulier puisque l'indéterminée est alors de niveau 1 et avons remarqué qu'à cause des contrôleurs, les relations $\rightarrow_{\mathcal{A}}$ et $\vdash_{\mathcal{A}}$ n'étaient définies que sur des termes ne contenant pas d'indéterminées de niveau 1. De plus, si $k = 1$, la condition* (**Q1**) *s'écrit $(q_0, a_2[{a_1}^n\Omega_1], q_0) \rightarrow_{\mathcal{B}}^{*} (q_0, \mathrm{A}_2[\Omega_1], q_0)^{s(n)}$ et on voit qu'il n'est plus possible d'insérer le séparateur B_1 entre a_1 et Ω_1 nécessaire à la preuve de la version forte du lemme.*

4. *La construction donnée pour la version faible est complètement indépendante des contrôleurs choisis, et il n'est pas nécessaire de connaître leur valeur pour effectuer la construction. Pour la version forte, la construction* effective *de l'automate nécessite de savoir calculer la valeur de* $\chi_{\vec{N}}(0)$.

Proposition 11.5.4 (Somme). *Si* $s,t \in \mathbb{S}^{\vec{N}}_{k+1}$, $k \geq 1$ *alors* $s+t \in \mathbb{S}^{\vec{N}}_{k+1}$.

Preuve : Soit $\mathcal{A}, \mathcal{A}'$ deux k-ACD calculant respectivement s and t. Supposons

$$q_0 a_{k+1}[T_{k,2}[{a_1}^n]]q_0 \rightarrow^*_{\mathcal{A}} \alpha^{s(n)} \text{ et } q_0 b_{k+1}[T_{k,2}[{a_1}^n]]q_0 \rightarrow^*_{\mathcal{A}'} \alpha^{t(n)}.$$

Il suffit de construire $\mathcal{B} \in k\text{-ACD}^{\vec{N}}$ produisant le calcul suivant : partant de l'état total $q_0 c_{k+1}[T_{k,2}[{a_1}^n]]$, (où c_{k+1} est un nouveau symbole) en effectuant l'instruction $\text{change}_{b_{k+1}}$ suivie de $\text{push}_{a_{k+1}}$, l'automate aboutit en $q_0 a_{k+1}[T_{k,2}[{a_1}^n]]b_{k+1}[T_{k,2}[{a_1}^n]]$, où q_0 est l'état de départ (et d'arrivée) commun des automates $\mathcal{A}$ et $\mathcal{A}'$. Il mime ainsi le comportement de $\mathcal{A}$ sur $q_0 a_{k+1}[T_{k,2}[{a_1}^n]]$, puis celui de $\mathcal{A}'$ sur $q_0 b_{k+1}[T_{k,2}[{a_1}^n]]$, et termine en q_0.

□

Proposition 11.5.5 (Produit ordinaire). *Si* $s,t \in \mathbb{S}^{\vec{N}}_{k+1}$, $k \geq 2$ *alors* $f \odot g \in \mathbb{S}_{k+1}$.

Construction : En utilisant le lemme 11.5.2, nous obtenons (après un choix approprié d'ensembles d'états et d'alphabets de piles) deux $(k+1)$-ACD$^{\vec{N}}$, $\mathcal{A}$ et $\mathcal{A}'$, vérifiant les conditions :
(**Q1**) $\forall \Omega_k \in \mathcal{I}_k, (q_0, a_{k+1}[a_k[\ldots[a_2[{a_1}^n]]\ldots]\Omega_k], q_0) \rightarrow^*_{\mathcal{A}} (q_0, \mathrm{A}_{k+1}[\Omega], q_0)^{f(n)}$.
(**Q1'**) $\forall \Omega_k \in \mathcal{I}_k, (q_0, \mathrm{A}_{k+1}[a_k[\ldots[a_2[{a_1}^n]]\ldots]\Omega_k], q_0) \rightarrow^*_{\mathcal{A}'} (q_0, \mathrm{B}_{k+1}[\Omega], q_0)^{g(n)}$.
(**Q2**) Δ n'a pas de membre gauche de la forme $(q_0, \varepsilon, \varepsilon)$.
(**Q2'**) Δ' n'a pas de membre gauche de la forme $(q_0, \varepsilon, \varepsilon)$.
(**Q3**) Δ n'a pas de membres gauche de la forme $(q_0, \varepsilon, \mathrm{A}_{k+1} \cdot w)$.
(**Q3'**) Δ' n'a pas de membre gauche de la forme $(q_0, \varepsilon, \mathrm{B}_{k+1} \cdot w)$.
(**Q4**) $Q \cap Q' = \{q_0\}$.
(**Q5**) $\forall i \in [1,k]$, $A_i \cap {A_i}' = \{a_i\}$ et $A_{k+1} \cap {A_{k+1}}' = \{\mathrm{A}_{k+1}\}$.
Nous construisons

$$\mathcal{B} = (Q \cup Q', \{\alpha\}, (B_1, \ldots, B_{k+1}), \vec{N}, \Delta \cup \Delta' \cup \Delta'', q_0)$$

où $B_i = A_i \cup A'_i$ pour $1 \leq i \leq k$ et $B_{k+1} = A_{k+1} \cup A'_{k+1} \cup \{b_{k+1}\}$ et Δ'' est l'union pour $\vec{o} \in \{0,1\}^{|\vec{N}|}$ des transitions suivantes :

(1) $\Delta''(q_0, \varepsilon, b_{k+1}a_k \cdots a_2, \vec{o}) = (\text{push}_{a_k} \text{change}_{a_{k+1}}, q_0)$

(2) $\Delta''(q_0, \alpha, \mathrm{B}_{k+1}, \vec{o}) = (\text{pop}_{k+1}, q_0)$

Preuve :
Déterminisme : Considérons (q_1, ε, w_1) membre gauche d'une règle de Δ, et (q_2, ε, w_2) membre gauche d'une règle de Δ'. Chacune peut-être appliquée a un même état total seulement si $q_1 = q_2$ et $w_1 = w_2$. Dans ce cas, par (**Q4**) $q_1 = q_2 = q_0$ et par (**Q5**) $w_1 = w_2 = \mathrm{A}_{k+1}w$. Mais la condition (**Q3**) rend impossibles de tels membres gauches pour Δ. Donc $\Delta \cup \Delta'$ est déterministe. L'ajout des transitions (1) ne brise pas ce déterministe puisque b_{k+1} est un nouveau symbole, enfin par (**Q3'**), une transition (2) n'est compatible avec aucune transition de Δ', et par (**Q5**) avec aucune transition de Δ. L'automate est donc déterministe.

De plus, par (**Q2**) et (**Q2'**), l'automate $\mathcal{B}$ vérifie la condition (**H2**) de la définition 11.2.1 : il n'y a aucune transitions ayant pour membre gauche $(q_0, \varepsilon, \varepsilon)$ ou $(q_0, \alpha, \varepsilon)$.

Calcul de la suite : Montrons maintenant que l'automate calcule $f \odot g$. Pour tout $n \geq 0$, les dérivations suivantes sont valides :

$$\begin{array}{rcll}
(q_0 b_{k+1}[T_{k,2}[{a_1}^n]]q_0) & \rightarrow_{\mathcal{A}} & (q_0 a_{k+1}[T_{k,2}[{a_1}^n]T_{k,2}[{a_1}^n]]q_0) & (\text{par } (1)) \\
& \rightarrow^*_{\mathcal{A}} & (q_0 \mathrm{A}_{k+1}[T_{k,2}[{a_1}^n]]q_0)^{f(n)} & (\text{par } (\mathbf{Q1})) \\
& \rightarrow^*_{\mathcal{A}} & (q_0 \mathrm{B}_{k+1}[\varepsilon]q_0)^{f(n)\cdot g(n)} & (\text{par } (\mathbf{Q1'})) \\
& \rightarrow^*_{\mathcal{A}} & \alpha^{f(n)\cdot g(n)} & (\text{par } (2))
\end{array}$$

□

Proposition 11.5.6. *Si $f \in \mathbb{S}^{\vec{N}}_{k+1}$, $k \geq 2$, alors la suite g définie par $g(0) = c \geq 1$ et $g(n+1) = f(n).g(n)^d$, pour $d \geq 1$, appartient à $\mathbb{S}^{\vec{N}}_{k+1}$.*

Preuve : Il existe un automate $\mathcal{A}_1 = (Q, \{\alpha\}, (A_1, \ldots, A_{k+1}), \vec{N}, \Delta_1, q_0) \in k\text{-}\mathsf{ACD}$ vérifiant les conditions $(\mathbf{Q1}(a_{+1} \cdots a_1, \mathrm{A}_{k+1})), (\mathbf{Q2}), (\mathbf{Q3}(\mathrm{A}_{k+1}))$ établies dans le lemme 11.5.2. Nous considérons l'automate

$$\mathcal{A} = (Q, \{\alpha\}, (B_1, \ldots, B_{k+1}), \vec{N}, \Delta, q_0) \in k\text{-}\mathsf{AC}$$

où $B_{k+1} = A_{k+1} \cup \{d_k\}$, $B_i = A_i$ pour tout $1 \leq i \leq k$ et Δ est l'union de Δ_1 avec les nouvelles transitions suivantes :

pour tout $\vec{o} \in \{0,1\}^{|\vec{N}|}$,

(0.1) $\Delta(q_0, \varepsilon, d_{k+1}a_k \cdots a_2 a_1, \vec{o}) = (\mathrm{pop}_1(\mathrm{push}_{a_k})^d \,\mathrm{change}_{a_{k+1}}, q_0)$

(0.2) $\Delta(q_0, \varepsilon, d_{k+1}a_k \cdots a_2, \vec{o}) = (\mathrm{pop}_k(\mathrm{push}_{d_{k+1}})^{c-1}, q_0)$

(1) $\Delta(q_0, \varepsilon, \mathrm{A}_{k+1}a_k \cdots a_2 a_1, \vec{o}) = \Delta(q_0, \varepsilon, \mathrm{A}_{k+1}a_k \cdots a_2, \vec{o}) = (\mathrm{change}_{d_{k+1}}, q_0)$

(2) $\Delta(q_0, \alpha, d_{k+1}, \vec{o}) = (\mathrm{pop}_{k+1}, q_0)$

Cet automate est déterministe : Δ_1 est une fonction de transition déterministe et par condition (**Q2**), les transitions (1) ne brisent pas le déterminisme. De plus, les transitions (0.i) et (2) sont incompatibles avec toute transition de Δ_1 (puisque d_{k+1} est un nouveau symbole), et ne peuvent pas interférer les unes avec les autres.
De façon à montrer que $\mathcal{A}$ calcule g, nous énumérons des dérivations de base intéressantes :
Règles d'initialisation, (I0) :
en utilisant les transitions (0.1),

$$(q_0 d_{k+1}[T_{k,2}[{a_1}^{n+1}]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 a_{k+1}[(T_{k,2}[{a_1}^n])^{d+1}\Omega_k]q_0)$$

Règles d'initialisation, (I0') ; et par les transitions (0.2) et la règle de décomposition,

$$(q_0 d_{k+1}[T_{k,2}[\varepsilon]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 d_{k+1}[\Omega_k]q_0)^c$$

Règles de calcul de f, (C1) :
Par (**P1**),

$$(q_0 a_{k+1}[T_{k,2}[{a_1}^n]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 \mathrm{A}_{k+1}[\Omega_k]q_0)^{f(n)}$$

Règle de recollement, (R2) : en utilisant les transitions (1), pour tout $n \geq 0$

$$(q_0 \mathrm{A}_{k+1}[T_{k,2}[{a_1}^n]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 d_{k+1}[T_{k,2}[{a_1}^n]\Omega_k]q_0)$$

Règle de terminaison, (T3) : en utilisant les transitions (2),

$$(q_0 d_{k+1}[\varepsilon]q_0) \rightarrow_{\mathcal{A}} \alpha$$

Montrons par induction sur $n \geq 0$, la propriété $\mathbf{P}(n)$ suivante :

$$(q_0 d_{k+1}[T_{k,2}[{a_1}^n]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 d_{k+1}[\Omega_k]q_0)^{g(n)}$$

Base : La règle d'initialisation (I0') prouve $\mathbf{P}(0)$.
Pas d'induction : Considérons la dérivation :

$$\begin{array}{rll} (q_0 d_{k+1}[T_{k,2}[{a_1}^{n+1}]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} & (q_0 a_{k+1}[(T_{k,2}[{a_1}^n])^{d+1}\Omega_k]q_0) & \text{(par la règle (I0))} \\ \rightarrow^*_{\mathcal{A}} & (q_0 \mathrm{A}_{k+1}[(T_{k,2}[{a_1}^n])^{d}\Omega_k]q_0)^{f(n)} & \text{(par } (C1)\text{)} \\ \rightarrow^*_{\mathcal{A}} & (q_0 d_{k+1}[(T_{k,2}[{a_1}^n])^{d}\Omega_k]q_0)^{f(n)} & \text{(par la règle (R2))} \end{array} \quad (29)$$

En appliquant d fois $\mathbf{P}(n)$ (avec les substitutions adéquates de l'indéterminée Ω_k), nous obtenons :

$$(q_0 d_{k+1}[(T_{k,2}[{a_1}^n])^{d}\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 d_{k+1}[\Omega_k]q_0)^{g(n)^d} \quad (30)$$

La composition des dérivations (29) et (30) donne :

$$(q_0 d_{k+1}[T_{k,2}[{a_1}^{n+1}]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 d_{k+1}[\Omega_k]q_0)^{f(n)\cdot g(n)^d} = (q_0 d_{k+1}[\Omega_k]q_0)^{g(n+1)}$$

La propriété $\mathbf{P}(n+1)$ est donc vrai, d'où $\mathbf{P}(n)$ vraie pour tout $n \geq 0$.
En appliquant la règle de terminaison (T3) à $\mathbf{P}(n)$, nous déduisons que, pour tout $n \geq 0$,

$$(q_0 d_{k+1}[T_{k,2}[{a_1}^n]]q_0) \rightarrow^*_{\mathcal{A}} \alpha^{g(n)}$$

□

Notons que par les proposition 11.5.4 et 11.5.5, pour tout $k \geq 3$, $(\mathbb{S}^{\vec{N}}_k, +, \cdot)$ est un semi-anneau. Nous notons $P(n, X_1, \ldots, X_p)$ un élément du semi-anneau $\mathbb{S}_k[X_1, \ldots, X_p]$ pour souligner le fait que les coefficients de P sont des fonctions de l'argument n.

Proposition 11.5.7. *Soient $k \geq 2$ et $P_i(n, X_1, \ldots, X_p)$, $1 \leq i \leq p$ des polynômes à coefficients dans $\mathbb{S}^{\vec{N}}_{k+1}$. Considérons les suites u_i, pour $1 \leq i \leq p$, définies par*
$u_i(n+1) = P_i(n, u_1(n), \ldots, u_p(n))$, et $u_i(0) = \gamma_i$. Alors $u_1 \in \mathbb{S}^{\vec{N}}_{k+1}$.

Esquisse de preuve : Le principe utilisé ici est le même que celui exposé dans la preuve de la proposition 11.5.6, étendu à plusieurs inconnues. Supposons que pour tout $1 \leq i \leq p$,

$$P_i(X_1, \ldots X_p) = \sum_{j=0}^{\nu_i} u_{i,j}(n) X_1^{d_{i,j,1}} \ldots X_p^{d_{i,j,p}}.$$

En utilisant le lemme 11.5.2, nous pouvons supposer que chaque coefficient $u_{i,j}(n)$ est calculé par automate

$$\mathcal{A}_{i,j} = (Q_{i,j}, \emptyset, (A_{1,i,j}, \ldots, A_{k,i,j}), \vec{N}, \Delta_{i,j}, q_0) \in (k+1)\text{-}\mathsf{ACD}^{\vec{N}}$$

vérifiant les conditions (**Q1**($a_{k+1,i,j}a_k\cdots a_1, \mathrm{A}_{k+1,i,j}$)), (**Q2**) et (**Q3**($\mathrm{A}_{k+1,i,j}$)) définies dans le lemme 11.5.2. Par un renommage des états et des alphabets de piles, nous obtenons pour tous couples $(i,j)\neq(i',j')$:

$$Q_{i,j}\cap Q_{i',j'}=\{q_0\};\ \ \forall l\in[1,k],\ A_{l,i,j}\cap A_{l,i',j'}=\{a_i\}\text{ et }A_{k+1,i,j}\cap A_{k+1,i',j'}=\emptyset$$

Supposons donné un automate $\mathcal{A}=(Q,\{\alpha\},(A_1,\ldots,A_k),\vec{N},\Delta,q_0)\in(k+1)\text{-}\mathsf{ACD}^{\vec{N}}$ tel que pour tout (i,j), Q contient tous les $Q_{i,j}$, Δ contient tous les $\Delta_{i,j}$, et pour tout ℓ, A_ℓ contient tous les $A_{l,i,j}$. Supposons de plus que A_k contient les nouveaux symboles $\mathrm{U}_{k,1},\mathrm{U}_{k,2},\ldots,\mathrm{U}_{k,p}$ et A_{k+1} contient le nouveau symbole a_{k+1}. Supposons enfin que les transitions permettent les dérivations de base suivantes :
Règles d'initialisation :

$$(q_0a_{k+1}[\mathrm{U}_{k,i}[T_{k-1}[{a_1}^{n+1}]]\Omega_k]q_0)\to^*_{\mathcal{A}}\prod_{j=0}^{\nu_i}(q_0a_{k+1,i,j}[(T_{k,2}[{a_1}^n])^2\Omega_k]q_0)$$

et

$$(q_0a_{k+1}[\mathrm{U}_{k,i}[T_{k-1,2}[\varepsilon]]\Omega]q_0)\to^*_{\mathcal{A}}(q_0a_{k+1}[\Omega_k]q_0)^{\gamma_i}$$

Règles de coefficients :

$$(q_0a_{k+1,i,j}[T_{k,2}[{a_1}^n]\Omega_k]q_0)\to^*_{\mathcal{A}_{i,j}}(q_0\mathrm{A}_{k+1,i,j}[\Omega_k]q_0)^{u_{i,j}(n)}$$

(il s'agit juste de la condition (**Q1**) pour l'automate $\mathcal{A}_{i,j}$)
Règles de recollement : pour tout $n\geq 0$, la règle de recollement est :

$$(q_0\mathrm{A}_{k+1,i,j}[T_{k,2}[{a_1}^n]\Omega_k]q_0)\to^*_{\mathcal{A}}(q_0a_{k+1}[(\prod_{\ell=1}^{p}(\mathrm{U}_{k,\ell}[T_{k-1,2}[{a_1}^n]])^{d_{i,j,\ell}})\Omega_k]q_0)$$

Règle de terminaison :

$$(q_0a_{k+1}[\varepsilon]q_0)\to_{\mathcal{A}}\alpha$$

Considérons la propriété $\mathbf{P}(n)$ définie par :

$$\forall i\in[1,p],\ (q_0a_{k+1}[\mathrm{U}_{k,i}[T_{k-1,2}[{a_1}^n]]\Omega_k]q_0)\to^*_{\mathcal{A}}(q_0a_{k+1}[\Omega_k]q_0)^{u_i(n)}.$$

La propriété $\mathbf{P}(\mathbf{n})$ se vérifie facilement par induction sur n : en appliquant successivement les règles d'initialisation, de coefficients, et de recollement, nous obtenons la dérivation suivante :

$$(q_0a_{k+1}[\mathrm{U}_{k,i}[T_{k-1,2}[{a_1}^{n+1}]]\Omega_k]q_0)\to^*_{\mathcal{A}}\prod_{j=0}^{\nu_i}(q_0a_{k+1}[(\prod_{\ell=1}^{p}(\mathrm{U}_{k,\ell}[T_{k-1,2}[{a_1}^n]])^{d_{i,j,\ell}})\Omega_k]q_0)^{u_{i,j}(n)}$$

En appliquant alors l'hypothèse $\mathbf{P}(n)$, nous obtenons $\mathbf{P}(n+1)$. L'application de la règle de terminaison à $\mathbf{P}(n)$ prouve alors que cet automate calcule bien la suite u_i.

En s'appuyant sur les propriétés de normalisation (**Q2**) et (**Q3**), il est possible d'ajouter des transitions à l'union des $\Delta_{i,j}$ de façon à ce que toutes ces règles soient rendues valides et que l'automate $\mathcal{A}$ reste déterministe :
- les variables des membres gauches des règles données ci-dessus sont distinctes
- il suffit donc de décomposer chacune de ces règles en une suite finie de mouvements élémentaires, utilisant des ensembles d'états disjoints pour les transitions intermédiaires des différentes règles, pour obtenir un tel automate *déterministe*. □

Proposition 11.5.8. *Soit $f \in \mathbb{S}^{\vec{N}}_{k+1}, g \in {\mathbb{S}_k}^{\vec{N}}$, $k \geq 3$, alors la suite h définie pour tout $n \geq 0$ par $h(n) = f(n)^{g(n)}$ appartient à $\mathbb{S}^{\vec{N}}_{k+1}$.*

Preuve : Procédons comme dans la preuve de la proposition 11.5.2 : nous exposons dans une première étape, une liste de dérivations particulières (que nous appelons "règles") et nous prouvons que ces règles sont suffisantes pour calculer la suite h ; dans une seconde étape, nous expliquons comment construire un automate déterministe rendant ces règles valides.

Première étape :

Soit $\mathcal{A} = (Q, \{\alpha\}, (A_1, \ldots, A_{k+1}), \vec{N}, \Delta, q_0)$ un $(k+1)$-ACD, avec $A_{k+1} \supseteq \{a_{k+1}, \mathrm{A}_{k+1}, b_{k+1}\}$, et $A_k \supseteq \{a_k, b_k, \mathrm{B}_k\}$ et pour tout $i \in [1, k-1]$, $A_i \supseteq \{a_i\}$.

Définissons le k-ACD : ${a_{k+1}}^{-1} \cdot \mathcal{A} = (Q, \{\alpha\}, (A_1, \ldots, A_k), \vec{N}, {a_{k+1}}^{-1} \cdot \Delta, q_0)$, où

$${a_{k+1}}^{-1} \cdot \Delta = \{(q, \alpha_\varepsilon, w, \vec{o}, q', \mathrm{instr}) \mid (q, \alpha_\varepsilon, a_{k+1}w, \vec{o}, q', \mathrm{instr}) \in \Delta \text{ et } \mathrm{instr} \in \mathcal{I}nstr_k\}.$$

Supposons que $\mathcal{A}$ permette les dérivations de base suivantes (où Ω_i est une indéterminée de niveau i) :

Règle d'initialisation, (I0) :

$$(q_0 b_{k+1}[T_{k,2}[{a_1}^n]]q_0) \rightarrow^*_{\mathcal{A}} (q_0 \mathrm{A}_{k+1}[b_k[T_{k-1,2}[{a_1}^n]\ T_{k-1,2}[{a_1}^n]]]q_0)$$

Calcul de f, (C1) :

$$(q_0 a_{k+1}[T_{k,2}[{a_1}^n]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 \mathrm{A}_{k+1}[\Omega_k]q_0)^{f(n)}$$

Calcul de g, (C2) :

$$(r_0 b_k[T_{k-1,2}[{a_1}^n]\Omega_{k-1}]r_0) \rightarrow^*_{{a_{k+1}}^{-1}\cdot\mathcal{A}} (r_0 \mathrm{B}_k[\Omega_{k-1}]r_0)^{g(n)}$$

Règles de recollement, (R12) : $\forall \omega_k \in k$-Pile,

$$(q_0 \mathrm{A}_{k+1}[\omega_k]q_0) \rightarrow_{\mathcal{A}} (r_0 a_{k+1}[\omega_k]q_0)$$

Règles de recollement, (R21) : $\forall \omega \in (k-1)$-Pile,

$$(r_0 a_{k+1}[\mathrm{B}_k[\omega_{k-1}]\Omega_k]q_0) \rightarrow_{\mathcal{A}} (q_0 a_{k+1}[a_k[\omega_{k-1}]\Omega_k]q_0)$$

Règles de recollement, ($\mathrm{R}^{(0)}21$) :

$$(r_0 a_{k+1}[\varepsilon]q_0) \rightarrow_{\mathcal{A}} (q_0 \mathrm{A}_{k+1}[\varepsilon]q_0)$$

Règles de terminaison, (T3) :

$$(q_0 \mathrm{A}_{k+1}[\varepsilon]q_0) \rightarrow_{\mathcal{A}} \alpha.$$

Les règles de recollement (Rij) permettent de connecter le fin d'un calcul (Ci) avec le début d'un calcul (Cj). La règle de recollement spéciale ($\mathrm{R}^{(0)}21$) traite le cas où le calcul (C2) résulte en le nombre 0, amenant à la valeur $f(n)^0 = 1$ (nous adoptons la convention $0^0 = 1$ dans la définition de $h = f^g$).

Prouvons par induction sur $i \geq 0$ la propriété suivante $\mathbf{P}(i)$:

pour tout $\omega_i \in k$-NPile$(A_1, \ldots, A_k)$, si

$$(r_0 \omega_i r_0) \rightarrow^*_{{a_{k+1}}^{-1}\mathcal{A}} (r_0 \mathrm{B}_k[T_{k-1,2}[{a_1}^n]]r_0)^i \tag{31}$$

alors

$$(q_0 \mathrm{A}_{k+1}[\omega_i]q_0) \rightarrow^*_{\mathcal{A}} (q_0 \mathrm{A}_{k+1}[\varepsilon]q_0)^{f(n)^i}. \tag{32}$$

Base : $i = 0$
Supposons que (31) est vérifié. La dérivation suivante est alors valide :

$$\begin{array}{lll} & (q_0 \mathrm{A}_{k+1}[\omega_0]q_0) & \\ \rightarrow_{\mathcal{A}} & (r_0 a_{k+1}[\omega_0]q_0) & (\text{ par règle (R12)}) \\ \rightarrow^*_{\mathcal{A}} & (r_0 a_{k+1}[\varepsilon]q_0) & (\text{ par hypothèse (31) et définition de } {a_{k+1}}^{-1} \cdot \mathcal{A}) \\ \rightarrow^*_{\mathcal{A}} & (q_0 \mathrm{A}_{k+1}[\varepsilon]q_0) & (\text{ par règle } (\mathrm{R}^{(0)}21) \text{ }). \end{array}$$

Pas d'induction :
Supposons que l'hypothèse (31) est vérifiée pour $i+1$ et que $\mathbf{P}(i)$ est vrai. En utilisant les propriétés de la relation de dérivation (lemme 2), nous pouvons traduire l'hypothèse (31) de la façon suivante :
il existe $\omega_i \in k\text{-}\mathsf{NPile}(A_1, \ldots, A_k)$ tel que

$$(r_0, \varepsilon, \omega_{i+1}) \vdash^*_{{a_{k+1}}^{-1} \cdot \mathcal{A}} (r_0, \varepsilon, \mathrm{B}_k[T_{k-1,2}[{a_1}^n]]\omega_i) \text{ et } (r_0 \omega_i r_0) \rightarrow^*_{{a_{k+1}}^{-1} \cdot \mathcal{A}} (r_0 \mathrm{B}_k[T_{k-1,2}[{a_1}^n]]r_0)^i$$

Nous obtenons alors la dérivation :

$$\begin{array}{lll} (q_0 \mathrm{A}_{k+1}[\omega_{i+1}]q_0) \rightarrow^*_{\mathcal{A}} & (r_0 a_{k+1}[\omega_{i+1}]q_0) & (\text{ par règle (R12)}) \\ \rightarrow^*_{\mathcal{A}} & (r_0 a_{k+1}[\mathrm{B}_k[T_{k-1,2}[{a_1}^n]]\omega_i]q_0) & (\text{ par la traduction ci-dessus}) \\ \rightarrow^*_{\mathcal{A}} & (q_0 a_{k+1}[a_k[T_{k-1,2}[{a_1}^n]]\omega_i]q_0) & (\text{ par règle (R21)}) \\ \rightarrow^*_{\mathcal{A}} & (q_0 \mathrm{A}_{k+1}[\omega_i]q_0)^{f(n)} & (\text{ par (C1)}) \end{array} \tag{33}$$

En composant cette dérivation avec $\mathbf{P}(i)$, nous obtenons :

$$(q_0 \mathrm{A}_{k+1}[\omega_{i+1}]q_0) \rightarrow^*_{\mathcal{A}} (q_0 \mathrm{A}_{k+1}[\varepsilon]q_0)^{f(n)^{i+1}}$$

(fin de l'induction).
Considérons $\omega = b_k[T_{k-1,2}[{a_1}^n]T_{k-1,2}[{a_1}^n]]$. D'après la règle (C2), ω vérifie l' hypothèse (31) pour l'entier $i = g(n)$. Ainsi, par $\mathbf{P}(i)$,

$$(q_0 \mathrm{A}_{k+1}[b_k[T_{k-1,2}[{a_1}^n]T_{k-1,2}[{a_1}^n]]]q_0) \rightarrow^*_{\mathcal{A}} (q_0 \mathrm{A}_{k+1}[\varepsilon]q_0)^{f(n)^{g(n)}}. \tag{34}$$

Finalement, en appliquant la règle d'initialisation, la dérivation (34), puis la règle de terminaison (T3), nous obtenons

$$(q_0, b_{k+1}[T_{k,2}[{a_1}^n]], q_0) \rightarrow^*_{\mathcal{A}} \alpha^{f(n)^{g(n)}}$$

Seconde étape
Nous avons donc montré qu'un automate réalisant les dérivations données précédemment calcule la suite h. Prouvons maintenant que nous savons construire un tel automate $\mathcal{A}$. La suite $f(n)$ est calculée par un $(k+1)$-ACD $\mathcal{A}_1 = (Q_1, \emptyset, (B_1, \ldots, B_{k+1}), \vec{N}, \Delta_1, q_0)$ vérifiant les conditions $(\mathbf{Q1}(a_{k+1} \cdots a_1, \mathrm{A}_{k+1}))$, $(\mathbf{Q2})$ et $(\mathbf{Q3}(\mathrm{A}_{k+1})$ établies dans le lemme 11.5.2. De même, la suite $g(n)$ est calculée par un k-ACD $\mathcal{A}_2 = (Q_2, \emptyset, (C_1, \ldots, C_k), \vec{N}, \Delta_2, r_0)$ vérifiant les mêmes conditions (pour les symboles de départ $b_k, a_{k-1}, \ldots, a_1$ et le symbole de terminaison B_k).

Supposons que $Q_1 \cap Q_2 = \emptyset$, $B_k \cap C_k = \emptyset$ et pour $i \in [1, k-1]$, $B_i \cap C_i = \{a_i\}$. Définissons $\mathcal{A} = (Q, \{\alpha\}, (A_1, \ldots, A_{k+1}), \vec{N}, \Delta, q_0)$ où

$$Q = Q_1 \cup Q_2, \;\; A_i = B_1 \cup C_i \text{ pour } i \in [1,k], \;\; A_{k+1} = B_{k+1} \cup \{b_{k+1}\},$$

et Δ est l'union de $\Delta_1 \cup (a_{k+1} \cdot \Delta_2)$ avec les règles suivantes :

(0) $\Delta(q_0, \varepsilon, b_{k+1}a_k \cdots a_2) = \Delta(q_0, \varepsilon, b_{k+1}a_k \cdots a_2 a_1) = (\text{change}_{b_k} \text{push}_{a_{k-1}} \text{change}_{\text{A}_{k+1}}, r_0)$

(1.2) $\Delta(q_0, \varepsilon, \text{A}_{k+1}w) = (\text{change}_{a_{k+1}}, r_0)$ pour tout $w \neq \varepsilon \in \text{top}(k\text{-}\mathsf{NPile}(A_1, \ldots, A_k))$,

(2.1.0) $\Delta(r_0, \varepsilon, a_{k+1}) = (\text{change}_{\text{A}_{k+1}}, q_0)$

(2.1) $\Delta(r_0, \varepsilon, a_{k+1}\text{B}_k w) = (\text{change}_{a_k}, q_0)$ pour tout $w \in \text{top}((k-1)\text{-}\mathsf{NPile}(A_1, \ldots, A_{k-1}))$

(3) $\Delta(q_0, \alpha, \text{A}_{k+1}) = (q_0, \text{pop}_{k+1})$.

L'automate $\mathcal{A}_1$ vérifiant (**Q3**(A_{k+1})), les transitions (1.2) et (3) n'introduisent pas de non-déterminisme, de même pour les transitions (2.1), puisque $\mathcal{A}_2$ vérifiant (**Q3**(B_k)) et $\text{B}_k \notin B_k$. Les transitions (2.1.0) utilise la paire (r_0, a_{k+1}) qui n'est pas utilisée dans $a_{k+1} \cdot \Delta_2$ par hypothèse (**Q2**) sur $\mathcal{A}_2$. Les transitions (0) utilisent un nouveau symbole. Donc $\mathcal{A}$ est déterministe et vérifie (**H2**).
Les transitions rendent valides les règles décrites dans la première étape : (C1) est vérifié par les transitions de Δ_1, (C2) est vérifié par celles de $a_{k+1} \cdot \Delta_2$, (R21) est vrai par transitions (2.1), ($\text{R}^{(0)}21$) est vrai par transition (2.1.0), (R12) est vrai par transitions (1.2) et (T3) est vrai par transition (3).
La proposition est donc prouvée.
□

Proposition 11.5.9. *Soit $k \geq 3$, et $P_i(n, X_1, \ldots, X_p)$, $1 \leq i \leq p$ des polynômes à coefficients dans $\mathbb{S}^{\vec{N}}_{k+1}$ et exposants dans $\mathbb{S}^{\vec{N}}_k$. Considérons les suites u_i, pour $1 \leq i \leq p$ définies par $u_i(n+1) = P_i(n, u_1(n), \ldots, u_p(n))$, et $u_i(0) = c_i$. Alors $u_1 \in \mathbb{S}^{\vec{N}}_{k+1}$.*

Esquisse de preuve : Le principe utilisé ici est le même que celui exposé dans la preuve de la proposition 11.5.8, étendu à plusieurs inconnues. Supposons que pour tout $1 \leq i \leq p$,

$$P_i(n, X_1, \ldots X_p) = \sum_{j=0}^{\nu_i} u_{i,j}(n) X_1^{d_{i,j,1}(n)} \ldots X_p^{d_{i,j,p}(n)}.$$

En utilisant le lemme 11.5.2, nous pouvons supposer que :

- chaque coefficient $u_{i,j}(n)$ est calculé par automate

$$\mathcal{A}_{i,j} = (Q_{i,j}, \emptyset, (A_{1,i,j}, \ldots, A_{k,i,j}), \vec{N}, \Delta_{i,j}, q_0) \in (k+1)\text{-}\mathsf{ACD}^{\vec{N}}$$

  vérifiant les conditions (**Q1**($a_{k+1,i,j}a_k \cdots a_1$)), (**Q2**) et (**Q3**($\text{A}_{k+1,i,j}$)) définies dans le lemme 11.5.2.
- chaque coefficient $d_{i,j,\ell}(n)$ est calculé par automate

$$\mathcal{B}_{i,j,\ell} = (Q_{i,j,\ell}, \emptyset, (B_{1,i,j,\ell}, \ldots, B_{k,i,j,\ell}), \vec{N}, \Delta_{i,j,\ell}, r_0) \in k\text{-}\mathsf{ACD}^{\vec{N}}$$

  vérifiant les conditions (**Q1**($b_{k,i,j,\ell}a_{k-1} \cdots a_1$)), (**Q2**) et (**Q3**($\text{B}_{k,i,j,\ell}$)) définies dans le lemme 11.5.2.

Par un renommage des états et des alphabets de piles, nous obtenons pour tous couples $(i,j) \neq (i',j')$:

$$Q_{i,j} \cap Q_{i',j'} = \{q_0\},\ \ \forall m \in [1,k],\ A_{m,i,j} \cap A_{m,i',j'} = \{a_i\} \text{ et } A_{k+1,i,j} \cap A_{k+1,i',j'} = \emptyset$$

et pour tous $(i,j,\ell) \neq (i',j',\ell')$:

$Q_{i,j,\ell} \cap Q_{i',j',\ell'} = \{r_0\},\ \ \forall m \in [1,k-1],\ B_{m,i,j,\ell} \cap B_{m,i',j',\ell'} = \{a_i\}$ et $B_{k+1,i,j,\ell} \cap B_{k+1,i',j',\ell'} = \emptyset$

enfin pour tous (i,j), (i',j',ℓ') :

$$Q_{i,j} \cap Q_{i',j',\ell'} = \emptyset,\ \forall m \in [1,k],\ A_{m,i,j} \cap B_{m,i',j',\ell'} = \emptyset.$$

Supposons donné un automate $\mathcal{A} = (Q, \{\alpha\}, (A_1, \ldots, A_k), \vec{N}, \Delta, q_0) \in (k+1)\text{-}\mathsf{ACD}^{\vec{N}}$ tel que pour tout (i,j,ℓ), Q contient tous les $Q_{i,j} \cup Q_{i,j,\ell}$, pour tout $m \in [1,k]$, A_m contient tous les $A_{m,i,j} \cup B_{m,i,j,\ell}$, A_{k+1} contient tous les $A_{k+1,i,j}$ et Δ contient tous les $\Delta_{i,j} \cup \Delta_{i,j,\ell}$. Supposons de plus que A_k contient les nouveaux symboles $\mathrm{U}_{k,1}, \mathrm{U}_{k,2}, \ldots, \mathrm{U}_{k,p}$ et A_{k+1} contient le nouveau symbole a_{k+1}. Supposons enfin que les transitions permettent les dérivations de base suivantes :
Règles d'initialisation :

$$(q_0 a_{k+1}[\mathrm{U}_{k,i}[T_{k-1,2}[{a_1}^{n+1}]]\Omega_k]q_0) \to^*_{\mathcal{A}} \prod_{j=1}^{\nu_i} (q_0 a_{k+1,i,j}[T_{k,2}[{a_1}^n][T_{k,2}[{a_1}^n]]\Omega_k]q_0)$$

Règles de coefficients :

$$(q_0 a_{k+1,i,j}[T_{k,2}[{a_1}^n]\Omega_k]q_0) \to^*_{\mathcal{A}} (q_0 \mathrm{A}_{k+1,i,j}[\Omega_k]q_0)^{u_{i,j}(n)}$$

(il s'agit juste de la condition (**Q1**) pour l'automate $\mathcal{A}_{i,j}$)
Règles de recollement, (R1) :

$$(q_0 \mathrm{A}_{k+1,i,j}[T_{k,2}[{a_1}^n]\Omega_k]q_0) \to^*_{\mathcal{A}} (q_0 a_{k+1}[(\prod_{\ell=1}^{p} b_{k,i,j,\ell}[T_{k-1,2}[{a_1}^n]T_{k-1,2}[{a_1}^n]])\Omega_k]r_0)$$

Règles de degrés :

$$(r_0 b_{k,i,j,\ell}[T_{k-1,2}[{a_1}^n]\Omega_{k-1}]r_0) \to^*_{{a_{k+1}}^{-1}.\mathcal{A}} (r_0 \mathrm{B}_{k,i,j,\ell}[\Omega_{k-1}]r_0)^{d_{i,j,\ell}(n)}$$

Règles de recollement, (R2) :

$$(r_0 a_{k+1}[\mathrm{B}_{k,i,j,\ell}[T_{k-1,2}[{a_1}^n]\Omega_k]q_0) \to^*_{\mathcal{A}} (q_0 a_{k+1}[\mathrm{U}_{k,\ell}[T_{k-1,2}[{a_1}^n]]\Omega_k]q_0)$$

Règle de terminaison :

$$(q_0 a_{k+1}[\varepsilon]q_0) \to_{\mathcal{A}} \alpha$$

Considérons la propriété $\mathbf{P}(n)$ définie par :

$$\forall i \in [1,p], (q_0 a_{k+1}[\mathrm{U}_{k,i}[T_{k-1,2}[{a_1}^n]]\Omega]q_0) \to^*_{\mathcal{A}} (q_0 a_{k+1}[\Omega]q_0)^{u_i(n)}.$$

La propriété $\mathbf{P}(n)$ peut être prouvée par induction sur n : en appliquant dans l'ordre les règles d'initialisation, de coefficients, de recollement (R1), de degrés, puis de recollement (R2), nous obtenons la dérivation suivante :

$$(q_0 a_{k+1}[\mathrm{U}_{k,i}[T_{k-1,2}[{a_1}^{n+1}]]\Omega_k]q_0) \to^*_{\mathcal{A}} \prod_{j=0}^{\nu_i} (q_0 a_{k+1}[(\prod_{\ell=1}^{p} (\mathrm{U}_{k,\ell}[T_{k-1,2}[{a_1}^n]])^{d_{i,j,\ell}(n)})\Omega_k]q_0)^{u_{i,j}(n)}$$

En appliquant alors l'hypothèse $\mathbf{P}(n)$, nous obtenons $\mathbf{P}(n+1)$. L'application de la règle de terminaison à $\mathbf{P}(n)$ prouve alors que cet automate calcule bien la suite u_i.

En s'appuyant sur les propriétés de normalisation (**Q2**) et (**Q3**), il est possible d'ajouter des transitions à l'union des $\Delta_{i,j}$, $\Delta_{i,j,\ell}$ de façon à ce que toutes ces règles soient rendues valides et que l'automate $\mathcal{A}$ reste déterministe :
- les variables des membres gauches des règles données ci-dessus sont distinctes
- il suffit donc de décomposer chacune de ces règles en une suite finie de mouvements élémentaires, utilisant des ensembles d'états disjoints pour les transitions intermédiaires des différentes règles, pour obtenir un tel automate *déterministe*. □

Proposition 11.5.10 (Produit de convolution). *Soit $f \in \mathbb{S}^{\vec{N}}_{k+1}$ et $g \in \mathbb{S}_k$, pour $k \geq 3$. Alors $f \times g \in \mathbb{S}^{\vec{N}}_{k+1}$ où $f \times g$ désigne le produit de convolution :*

$$(f \times g)(n) = \sum_{m=0}^{n} f(m) \cdot g(n-m) \text{ pour tout } n \in \mathbb{N}$$

Preuve : La difficulté principale est de définir pour tous $0 \leq m \leq n$, une $(k+1)$-Npile $\omega_{m,n-m}$ à partir de laquelle nous savons calculer $f(m) \cdot g(n-m)$, et de générer la suite des $\omega_{n,0}$, $\omega_{n-1,1}$, $\ldots, \omega_{0,n}$. Nous avons besoin cette fois de deux compteurs qui doivent pouvoir évoluer simultanément, il n'est donc pas possible de les placer tous les deux au niveau 1 de la pile. Le compteur a_1 de la suite f sera placé au niveau 1 comme usuellement, tandis que celui de la suite g sera noté b_2 et placé au niveau 2. Ceci explique le fait que les suites f et g appartiennent à deux classes de niveaux différents et que nous n'autorisons pas l'automate calculant g à être contrôlé, puisque les contrôles ne peuvent porter que sur le niveau 1 des piles.

Nous codons chaque couple $(m, n-m)$ par la 2-pile suivante :

$$\gamma_{m,n} = a_2[{a_1}^m]b_2[{a_1}^{m+1}] \cdots b_2[{a_1}^n],\ m \neq n,\ \gamma_{n,n} = a_2[{a_1}^n]$$

L'entier m est codé comme usuellement dans le premier atome, tandis que l'entier $n-m$ correspond à la longueur du suffixe $b_2[{a_1}^{m+1}] \cdots b_2[{a_1}^n]$. Nous calculons alors $f(m) \cdot g(n-m)$ en utilisant le même type d'argument que celui de la proposition 11.5.5 concernant le produit.
Première étape

Supposons donné $\mathcal{A} = (Q, \{\alpha\}, (A_1, \ldots, A_{k+1}), \vec{N}, \Delta, q_0) \in (k+1)\text{-}\mathsf{AC}^{\vec{N}}$ avec $A_1 = \{a_1\}$, $A_2 \supseteq \{a_2, b_2\}$, $A_i \supseteq \{a_i\}$ pour $i \in [3,k]$ et $A_{k+1} \supseteq \{a_{k+1}, \mathrm{A}_{k+1}\mathrm{B}_{k+1}\}$.

Comme précédemment, la lettre Ω_i désigne une indéterminée de niveau i. Les lettres a_2 et b_2 sont utilisées comme "compteurs" pour la suite g tandis que a_1 est le compteur utilisé pour la suite f. Nous notons encore

$$T_{k,3}[\Omega_2] = a_k[\cdots[a_3[\Omega_2]]\cdots]$$

Supposons que $\mathcal{A}$ permette les dérivations de base suivantes :
Calcul de f, (C1) : pour tout $n, m \geq 0$

$$(q_0 b_{k+1}[T_{k,3}[\gamma_{m,n}]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 \mathrm{B}_{k+1}[\Omega_k]q_0)^{f(m)}$$

Calcul de g, (C2) : pour tous $n \geq 0, m \in [0,n]$

$$(q_0 \mathrm{B}_{k+1}[T_{k,3}[\gamma_{n,m}]]q_0) \rightarrow^*_{\mathcal{A}} (q_0 \mathrm{A}_{k+1}[\varepsilon]q_0)^{g(m)}$$

Génération des paires, (G3) : pour tous $1 \leq m \leq n$

$$(q_0 a_{k+1}[T_{k,3}[\gamma_{m,n}]]q_0) \to^*_{\mathcal{A}} (q_0 a_{k+1}[T_{k,3}[\gamma_{m-1,n}]]q_0)(q_0 b_{k+1}[T_{k,3}[\gamma_{m,n}]T_{k,3}[\gamma_{m,n}]]q_0)$$

Génération de la paire initiale, (G30) : pour tout $n \geq 0$

$$(q_0 a_{k+1}[T_{k,3}[\gamma_{0,n}]]q_0) \to^*_{\mathcal{A}} (q_0 b_{k+1}[T_{k,3}[\gamma_{0,n}]T_{k,3}[\gamma_{0,n}]]q_0)$$

Règle de terminaison, (T4) :

$$(q_0 \mathrm{A}_{k+1}[\varepsilon]q_0) \to^*_{\mathcal{A}} \alpha.$$

Comme $\gamma_{n,n} = a_2[{a_1}^n]$, en appliquant itérativement (G3) puis (G30), nous obtenons :

$$(q_0 a_{k+1}[T_{k,2}[{a_1}^n]]q_0) \quad \to^*_{\mathcal{A}} \quad \prod_{m=0}^{n} (q_0 b_{k+1}[T_{k,3}[\gamma_{m,n}]T_{k,3}[\gamma_{m,n}]]q_0) \tag{35}$$

En partant de chaque facteur de ce produit nous dérivons :

$$\begin{aligned} (q_0 b_{k+1}[T_{k,3}[\gamma_{m,n}]T_{k,3}[\gamma_{m,n}]]q_0) &\to^*_{\mathcal{A}} \quad (q_0 \mathrm{B}_{k+1}[T_{k,3}[\gamma_{m,n}]]q_0)^{f(m)} \quad \text{(par règle (C1))} \\ &\to^*_{\mathcal{A}} \quad (q_0 \mathrm{A}_{k+1}[\varepsilon]q_0)^{g(n-m)\cdot f(m)} \quad \text{(par règle (C2))} \end{aligned} \tag{36}$$

En composant les dérivations (35) et (36), nous obtenons :

$$\begin{aligned} (q_0 a_{k+1}[T_{k,3}[a_2[{a_1}^n]]]q_0) \quad &\to^*_{\mathcal{A}} \quad (q_0 \mathrm{A}_{k+1}[\varepsilon]q_0)^{\sum_{m=0}^{n} g(n-m)\cdot f(m)} \\ &= \quad (q_0 \mathrm{A}_{k+1}[\varepsilon]q_0)^{(f\times g)(n)} \end{aligned}$$

Seconde étape :

Construisons maintenant un automate déterministe $\mathcal{A}$ réalisant ces dérivations. La suite $f(n)$ est calculée par un $(k+1)$-ACD $\mathcal{A}'_1$ vérifiant les conditions (**Q1**$(b_{k+1}a_k \cdots a_1, \mathrm{B}_{k+1})$, (**Q2**) et (**Q3**$(\mathrm{B}_{k+1})$) posées dans le lemme 11.5.2. Nous introduisons le nouveau symbole b_2, en ajoutant des transitions permettant de traiter b_2 comme un symbole de fond de pile. Nous obtenons ainsi un automate $\mathcal{A}_1 = (Q_1, \emptyset, (B_1, \ldots, B_{k+1}), \vec{N}, \Delta_1, q_0)$, où $B_{k+1} \supseteq \{b_{k+1}, \mathrm{B}_{k+1}\}$, $B_i \supseteq \{a_i\}$ pour $i \in [3,k]$, $B_2 \supseteq \{a_2, b_2\}$ et $B_1 = \{a_1\}$ vérifiant les conditions (C1), (**Q2**) et (**Q3**(B_{k+1})).

Par le lemme 11.5.1, la suite $g(n)$ est calculée par un k-ACD

$$\mathcal{A}'_2 = (Q'_2, \emptyset, (C_2, \ldots, C_{k+1}), \Delta'_2, q_0),$$

où $C_2 = \{b_2\}$, $C_i \supseteq \{a_i\}$ pour $i \in [3,k]$, $C_{k+1} \supseteq \{\mathrm{B}_{k+1}, \mathrm{A}_{k+1}\}$ et vérifiant la condition (**P1**$(q_0, \mathrm{B}_{k+1}a_k \cdots a_3 b_2, \mathrm{A}_{k+1})$), la condition (**P2**) et la condition (**P3**(A_{k+1})). Il est alors facile de transformer $\mathcal{A}_1$ en un $(k+1)$-ACD contrôlé $\mathcal{A}_2 = (Q_2, \{\alpha\}, (C_1, \ldots, C_{k+1}), \vec{N}, \Delta_2, q_0)$, avec $C_1 = \{a_1\}$, et vérifiant (C2) et (**P2**) et (**P3**).

Il suffit principalement de poser pour chaque $\Delta_2{}'(p, \varepsilon, w) = (\mathrm{instr}, q)$ tel que $|w| = k$,

$$\Delta_2(p, \varepsilon, w, \vec{o}) = (p, \varepsilon, wa_1) = (\mathrm{instr}^{+1}, q),\ \forall \vec{o} \in \{0,1\}^{|\vec{N}|}$$

et pour chaque $\Delta_2{}'(p, \varepsilon, w) = (\mathrm{instr}, q)$ tel que $|w| < k$,

$$\Delta_2(p, \varepsilon, w, \vec{o}) = (\mathrm{instr}^{+1}, q),\ \forall \vec{o} \in \{0,1\}^{|\vec{N}|}$$

avec pour tout $i \in [1,k]$, ${\rm pop}_i{}^{+1} = pop_{i+1}$, ${\rm push}_{i,a}{}^{+1} = {\rm push}_{i+1,a}$ et ${\rm change}_{i,a}{}^{+1} = {\rm change}_{i+1,a}$.

Le symbole a_1 est donc ignoré par l'automate et les contrôleurs n'ont aucune influence sur les transitions. Nous choisissons les alphabets de façon à ce que $B_2 \cap C_2 = \{a_2, b_2\}$, $B_i \cap C_i = \{a_i\}$ pour $i \in [3,k]$ et $B_{k+1} \cap C_{k+1} = \{{\rm B}_{k+1}\}$.
Définissons $\mathcal{A} = (Q, \{\alpha\}, (A_1, \ldots, A_{k+1}), \vec{N}, \Delta, q_0, \bot)$ où

$$Q = Q_1 \cup Q_2 \cup \{r_1\};\ \ A_i = B_i \cup C_i \text{ pour } i \in [1,k], \text{ et } A_{k+1} = B_{k+1} \cup C_{k+1};$$

Δ est l'union de $(\Delta_1 \cup \Delta_2)$ avec les transitions suivantes : $\forall \vec{o} \in \{0,1\}^{|\vec{N}|}$,

(3.1) $\Delta(q_0, \varepsilon, a_{k+1}a_k \cdots a_2 a_1, \vec{o}) = ({\rm push}_{a_k}\, {\rm change}_{b_{k+1}}\, {\rm push}_{a_{k+1}}\, {\rm pop}_k, r_1)$

(3.2) $\Delta(r_1, \varepsilon, a_{k+1}a_k \cdots a_2 a_1, \vec{o}) = ({\rm change}_{b_2}\, {\rm push}_{a_2}\, {\rm pop}_1, q_0)$

(30) $\Delta(q_0, \varepsilon, a_{k+1}a_k \cdots a_2, \vec{o}) = ({\rm push}_{a_k}\, {\rm push}_{b_{k+1}}, q_0)$

(4) $\Delta(q_0, \alpha, {\rm A}_{k+1}, \vec{o}) = ({\rm pop}_{k+1}, q_0)$.

Les automates initiaux $\mathcal{A}_i$ sont déterministes, et puisque $\mathcal{A}_2$ vérifie (**P2**(A_{k+1}), le nouvel automate $\mathcal{A}$ est également déterministe.
Les transitions sont choisies de façon à rendre valides les règles décrites dans la première étape : (Ci) est vraie par Δ_i, $(i = 1, 2)$, (G3) est valide par les transitions (3.j), (G30) est réalisé par les transitions (30), et (T4) par les transitions (4). Nous pouvons donc conclure que $\mathcal{A}$ calcule $f \times g$.

□

Le mode de génération de paires que nous avons utilisé oblige l'automate final à être de niveau au moins 4 (pour pouvoir obtenir $b_{k+1}[T_{k,3}[\gamma_{m,n}]T_{k,3}[\gamma_{m,n}]]$ à partir de $a_{k+1}[T_{k,3}[\gamma_{m,n}]]$) Cependant, la première copie est utilisée pour calculer $f(m)$ et nous n'avons donc pas besoin de $T_{k,3}[\gamma_{m,n}]$ tout entier, mais simplement de la valeur de m. Puisque pour la construction du système dérivant un calcul de g (dérivation (C2)), nous utilisons la version faible du lemme de normalisation, qui est valable pour des automates de niveau au moins 2, il est en fait possible de montrer la proposition pour $k = 2$ mais la preuve est légèrement différente.

Proposition 11.5.11 (Produit de convolution). *Soit $f \in \mathbb{S}_3{}^{\vec{N}}$ et $g \in \mathbb{S}_2$. Alors $f \times g \in \mathbb{S}_3{}^{\vec{N}}$ où $f \times g$ désigne le produit de convolution :*

$$(f \times g)(n) = \sum_{m=0}^{n} f(m) \cdot g(n-m) \text{ pour tout } n \in \mathbb{N}$$

Preuve : Nous utilisons ici les mêmes notations que dans la preuve de lemme précédent en ajoutant $\gamma'_{m,n} = b_2[{a_1}^{m+1}] \cdots b_2[{a_1}^{n}]$ (c'est à dire $\gamma_{m,n} = a_2[{a_1}^{m}]\gamma'_{m,n}$).
Première étape : Supposons que nous disposons d'un automate $\mathcal{A} \in k\text{-}\mathsf{AC}^{\vec{N}}$ pour lequel les dérivations suivantes sont valides :
Calcul de f, (C1) : pour tout $n \geq 0$

$$(q_0 b_3[a_2[{a_1}^{n}]\Omega_2]q_0) \rightarrow_{\mathcal{A}}^{*} (q_0 {\rm B}_2[\Omega_2]q_0)^{f(n)}$$

Calcul de g, (C2) : pour tous $n \geq 0, m \in [0,n]$

$$(q_0 {\rm B}_3[\gamma'_{m,n}]q_0) \rightarrow_{\mathcal{A}}^{*} (q_0 {\rm A}_3[\varepsilon]q_0)^{g(n-m)}$$

Génération des paires, (G3) : pour tous $1 \leq m \leq n$

$$(q_0 a_3[\gamma_{m,n}]q_0) \rightarrow^*_{\mathcal{A}} (q_0 a_3[\gamma_{m-1,n}]q_0)(q_0 b_3[\gamma_{m,n}]q_0)$$

Génération de la paire initiale, (G30) : pour tout $0 \leq n$

$$(q_0 a_3[\gamma_{0,n}]q_0) \rightarrow^*_{\mathcal{A}} (q_0 b_3[\gamma_{0,n}]q_0)$$

Règle de terminaison, (T4) :

$$(q_0 \mathrm{A}_3[\varepsilon]q_0) \rightarrow^*_{\mathcal{A}} \alpha.$$

À partir de telles règles, nous obtenons les dérivations suivantes :

$$\begin{array}{rcll}
(q_0 a_3[a_2[{a_1}^n]]q_0) \rightarrow^*_{\mathcal{A}} & \prod_{m=0}^{n}(q_0 b_3[\gamma_{m,n}]q_0) & \text{(par règles (G3), (G30))} \\
\rightarrow^*_{\mathcal{A}} & \prod_{m=0}^{n}(q_0 \mathrm{B}_3[\gamma'_{m,n}]q_0)^{f(m)} & \text{(par règle (C1))} \\
\rightarrow^*_{\mathcal{A}} & \prod_{m=0}^{n}(q_0 \mathrm{A}_3[\varepsilon]q_0) & \text{(par règle (C2))} \\
\rightarrow^*_{\mathcal{A}} & \alpha^{h(n)} & \text{(par règle (T4))}
\end{array}$$

Seconde étape :
Le lemme 11.5.2, permet d'obtenir les transitions rendant valide (C1), celles permettant (C2) sont obtenues comme dans la proposition précédente sauf qu'il n'est plus nécessaire d'effacer $a_2[{a_1}^m]$ avant de débuter le calcul. La dérivation (T4) est obtenue de façon évidente, enfin, le système permettant les règles de génération des paires est l'union pour $\vec{o} \in \{0,1\}^{|\vec{N}|}$ des transitions :

(3) $\Delta(q_0, \varepsilon, a_3 a_2 a_1, \vec{o}) = (\mathrm{change}_{b_3}\, \mathrm{push}_{a_3}\, \mathrm{change}_{b_2}\, \mathrm{push}_{a_2}\, \mathrm{pop}_1, q_0)$

(30) $\Delta(q_0, \varepsilon, a_3 a_2, \vec{o}) = (\mathrm{change}_{b_3}, q_0)$

Par un choix approprié des ensembles concrets d'états et d'alphabets de piles, nous obtenons donc un automate déterministe calculant $f \times g$.

□

Proposition 11.5.12 (Inverse de convolution). *Soit $g \in \mathbb{S}_k$, $k \geq 3$, et f la suite définie par $f(0) = 1$ et pour tout $n \geq 0$, $f(n+1) = \sum_{m=0}^{n} f(m) \cdot g(n-m)$. Alors $f \in \mathbb{S}_{k+1}$.*

Esquisse de preuve : Nous utilisons ici la même ligne directrice et les même notations que pour la proposition 11.5.10.
Première étape : Supposons donné un $(k+1)$-ACD $\mathcal{A} = (Q, \{\alpha\}, (A_1, \ldots, A_{k+1}), \Delta, q_0)$, avec $A_{k+1} \supseteq \{a_{k+1}, \mathrm{A}_{k+1}, b_{k+1}, \mathrm{B}_{k+1}\}$, $A_i \supseteq \{a_i\}$ pour $i \in [k,3]$, $A_2 \supseteq \{a_2, b_2, \mathrm{B}_2\}$ et $A_1 = \{a_1\}$, où le nouveau symbole B_2 joue le rôle de fond de pile pour l'automate calculant g. On appelle ici "pile bloquante" toute 2-pile ω de la forme $\mathrm{B}_2[\omega_1] \cdot \omega_2$, pour $\omega_i \in i\text{-}\mathsf{NPile}$ ou $\omega = \varepsilon$. Comme précédemment, nous codons le couple $(m, n-m)$ par la 2-pile

$$\gamma_{m,n} = a_2[a_1^m]b_2[a_1^{m+1}] \cdots b_2[{a_1}^n]$$

Les piles bloquantes seront utilisées pour construire des règles permettant de recoller chaque élément $\omega_{i,j}$ amenant au calcul de $f(i)g(j)$. Supposons que $\mathcal{A}$ permette les dérivations de base suivantes :

Calcul de g, (C2) : pour tous $\ell \geq 1$, $i_0, i_1, \ldots, i_\ell \geq 0$ et ω, pile bloquante

$$(q_0 b_{k+1}[T_{k,3}[a_2[{a_1}^{i_0}]b_2[{a_1}^{i_1}]\cdots b_2[{a_1}^{i_\ell}]\omega]\Omega_k]q_0 \to_{\mathcal{A}}^* (q_0 \mathrm{B}_{k+1}[\Omega_k]q_0)^{g(\ell)}$$

Génération des paires, (G3) : pour tout $n \geq 0$

$$(q_0 a_{k+1}[T_{k,3}[\gamma_{n+1,n+1}\Omega_2]]q_0) \to_{\mathcal{A}}^* \prod_{m=0}^{n} (q_0 b_{k+1}[(T_{k,3}[\gamma_{m,n}\Omega_2])^2]q_0)$$

Paires de départ, (G30) :

$$(q_0 a_{k+1}[T_{k,3}[\gamma_{0,0}\Omega_2]\Omega_k]q_0) \to_{\mathcal{A}}^* (q_0 \mathrm{B}_{k+1}[\Omega_k]q_0)$$

Règle de recollement, (R23) : pour tous $0 \leq m \leq n$, il existe une pile bloquante ω telle que,

$$(q_0 \mathrm{B}_{k+1}[T_{k,3}[\gamma_{m,n}\Omega_2]\Omega_k]q_0) \to_{\mathcal{A}}^* (q_0 a_{k+1}[T_{k,3}[\gamma_{m,m}\omega\Omega_2]\Omega_k]q_0)$$

Règle de terminaison, (T4) :

$$(q_0 \mathrm{B}_{k+1}[\varepsilon]q_0) \to_{\mathcal{A}}^* \alpha.$$

Prouvons par induction la propriété $\mathbf{P}(n)$ suivante : pour tout $0 \leq m \leq n$ et toute pile bloquante ω

$$(q_0 a_{k+1}[T_{k,3}[\gamma_{m,m}\omega]]q_0) \to_{\mathcal{A}}^* (q_0 \mathrm{B}_{k+1}[\varepsilon]q_0)^{f(m)}$$

Base : $\mathbf{P}(0)$ est prouvée par (D30), en substituant ω à Ω_2 et ε à Ω_k.
Pas d'induction :

$$\begin{aligned}
&(q_0 a_{k+1}[T_{k,3}[\gamma_{n+1,n+1}\omega]]q_0) \\
&\qquad (\text{par (G3)}) \quad \to_{\mathcal{A}}^* \quad \prod_{m=0}^{n} (q_0 b_{k+1}[(T_{k,3}[\gamma_{m,n}\omega])^2]q_0) \\
&\qquad (\text{par (C2)}) \quad \to_{\mathcal{A}}^* \quad \prod_{m=0}^{n} (q_0 \mathrm{B}_{k+1}[T_{k,3}[\gamma_{m,n}\omega]]q_0)^{g(n-m)} \\
&\qquad (\text{par (R23)}) \quad \to_{\mathcal{A}}^* \quad \prod_{m=0}^{n} (q_0 a_{k+1}[T_{k,3}[\gamma_{m,m}\omega_m]]q_0)^{g(n-m)} \\
&(\text{par hypothèse d'induction}) \quad \to_{\mathcal{A}}^* \quad \prod_{m=0}^{n} (q_0 \mathrm{B}_{k+1}[\varepsilon]q_0)^{f(m)\cdot g(n-m)} \\
&\qquad\qquad = \quad (q_0 \mathrm{B}_{k+1}[\varepsilon]q_0)^{f(n+1)}
\end{aligned}$$

(où toutes les ω_m sont des piles bloquantes). En utilisant (T4) nous obtenons finalement :

$$\forall n \in \mathbb{N}, (q_0 a_{k+1}[T_{k,2}[{a_1}^n]]q_0) \to_{\mathcal{A}}^* \alpha^{f(n)}.$$

Seconde étape : On peut construire un automate $\mathcal{A}_2$ réalisant (D2) et vérifiant aussi la condition (**P2**) du lemme 11.5.2. Les règles de départ et de terminaison peuvent être rendues valides par un ensemble de transitions, de façon similaire à celui utilisé dans la preuve de la proposition 11.5.10. Les règles de recollement (R23) sont obtenues grâce aux transitions suivantes :

$\Delta(q_0, \varepsilon, \mathrm{B}_{k+1}a_k \cdots a_2) = \Delta(q_0, \varepsilon, \mathrm{B}_{k+1}a_k \cdots a_2a_1) = (\mathrm{change}_{\mathrm{B}_2}\, \mathrm{push}_{a_2}\, \mathrm{change}_{a_{k+1}}, q_0)$.
Pour tout $\gamma_{m,n} = a_2[{a_1}^m]\omega$, nous obtenons la dérivation

$$(q_0, \mathrm{B}_{k+1}[T_{k,3}[\gamma_{m,n}]], q_0) \to_{\mathcal{A}} (q_0, a_{k+1}[T_{k,3}[a_2[{a_1}^m]\mathrm{B}_2[{a_1}^m]\omega]], q_0)$$

La règle (R23) est donc valide. □

Comme pour le cas de la convolution, il est possible de démontrer cette proposition pour $k = 3$, mais la preuve est légèrement différente.

Proposition 11.5.13 (convolution-inverse). *Soit $g \in \mathbb{S}_2$, et f la suite définie par $f(0) = 1$ et pour tout $n \geq 0$, $f(n+1) = \sum_{m=0}^{n} f(m) \cdot g(n-m)$. Alors $f \in \mathbb{S}_3$.*

Preuve : Nous utilisons ici les mêmes notations que dans la preuve de lemme précédent en ajoutant $\gamma'_{m,n} = b_2[{a_1}^{m+1}] \cdots b_2[{a_1}^n]$ (c'est à dire $\gamma_{m,n} = a_2[{a_1}^m]\gamma'_{m,n}$). Nous appelons encore pile bloquante toute 2-pile telle que $\mathrm{top}_2(\omega) = \mathrm{B}_2$. Supposons que nous disposons d'un automate $\mathcal{A} \in k\text{-}\mathsf{AC}^{\vec{N}}$ pour lequel les dérivations suivantes sont valides :
Règle d'initialisation, (I0) : pour tout $n \geq 0$

$$(q_0, d_3[a_2[{a_1}^n]], q_0) \to^*_{\mathcal{A}} (q_0, a_3[\gamma_{n,n}\mathrm{B}_2[{a_1}^n]], q_0)$$

Calcul de g, (C2) : pour tous $n \geq 0, m \in [1, n]$, pour toute pile bloquante ω

$$(q_0b_3[\gamma'_{m,n}\omega]q_0) \to^*_{\mathcal{A}} (q_0\mathrm{B}_3[\omega]q_0)^{g(n-m)}$$

Génération des paires, (G3) : pour tous $n \geq 0$

$$(q_0a_3[a_2[{a_1}^{n+1}]\Omega_2]q_0) \to^*_{\mathcal{A}} \prod_{m=0}^{n} (q_0a_3[a_2[{a_1}^m]\mathrm{B}_2[{a_1}^m]\gamma'_{m,n}\Omega_2]q_0)$$

Paire de départ, (G30) : pour tout $0 \leq n$

$$(q_0a_3[\gamma_{0,0}\Omega_2]q_0) \to^*_{\mathcal{A}} (q_0\mathrm{B}_3[\Omega_2]q_0)$$

Règles de terminaison, (T0) :

$$(q_0\mathrm{A}_3[\varepsilon]q_0) \to^*_{\mathcal{A}} \alpha.$$

(T1) :

$$(q_0\mathrm{B}_3[\mathrm{B}_2[\Omega_1]\Omega_2]q_0) \to^*_{\mathcal{A}} (q_0\mathrm{A}_3[\Omega_2]q_0)$$

Règle de recollement, (R4) : pour tout $\omega \neq \varepsilon \in 2\text{-}\mathsf{NPile}$

$$(q_0\mathrm{A}_3[\omega]q_0) \to^*_{\mathcal{A}} (q_0b_3[\omega]q_0)$$

À partir de ces règles, nous prouvons par induction sur $n \geq 0$ la propriété $\mathbf{P}(n)$ suivante :

$$(q_0a_3[a_2[{a_1}^n]\mathrm{B}_2[\Omega_1]\Omega_2]q_0) \to^*_{\mathcal{A}} (q_0\mathrm{A}_3[\Omega_2]q_0)^{f(n)}$$

Base : En appliquant (G30) puis (T1), nous obtenons

$$(q_0a_3[\gamma_{0,0}\mathrm{B}_2[\Omega_1]\Omega_2]q_0) \to^*_{\mathcal{A}} (q_0\mathrm{B}_3[\mathrm{B}_2[\Omega_1]\Omega_2]q_0) \to^*_{\mathcal{A}} (q_0\mathrm{A}_3[\Omega_2]q_0)$$

Puisque $\gamma_{0,0} = a_2[\varepsilon]$ et $f(0) = 1$, $\mathbf{P}(0)$ est prouvée.
Pas d'induction : Supposons $\mathbf{P}(n)$, pour $n \geq 0$,

$$\begin{aligned}
&(q_0 a_3[a_2[{a_1}^{n+1}]\mathrm{B}_2[\Omega_1]\Omega_2]q_0) \\
&\qquad (\text{par (G3)}) \quad \to_{\mathcal{A}}^* \quad \prod_{m=0}^{n} (q_0 a_3[a_2[{a_1}^m]\mathrm{B}_2[{a_1}^m]\gamma'_{m,n}\mathrm{B}_2[\Omega_1]\Omega_2]q_0) \\
&(\text{par hypothèse d'induction}) \quad \to_{\mathcal{A}}^* \quad \prod_{m=0}^{n} (q_0 \mathrm{A}_3[\gamma'_{m,n}\mathrm{B}_2[\Omega_1]\Omega_2]q_0)^{f(m)} \\
&\qquad (\text{par (R4)}) \quad \to_{\mathcal{A}}^* \quad \prod_{m=0}^{n} (q_0 b_3[\gamma'_{m,n}\mathrm{B}_2[\Omega_1]\Omega_2]q_0)^{f(m)} \\
&\qquad (\text{par (C2)}) \quad \to_{\mathcal{A}}^* \quad \prod_{m=0}^{n} (q_0 \mathrm{B}_3[\mathrm{B}_2[\Omega_1]\Omega_2]q_0)^{f(m)g(n-m)} \\
&\qquad (\text{par (T1)}) \quad \to_{\mathcal{A}}^* \quad \prod_{m=0}^{n} (q_0 \mathrm{A}_3[\Omega_2]q_0)^{f(m)g(n-m)} \\
&\qquad\qquad\qquad = \quad (q_0 \mathrm{A}_3[\Omega_2]q_0)^{f(n+1)}
\end{aligned}$$

Alors, en composant la règle d'initialisation, la dérivation ci-dessus, et la règle de terminaison (T0), nous obtenons pour tout $n \geq 0$:

$$\begin{aligned}
(q_0 d_3[a_2[{a_1}^n]]q_0) &\to_{\mathcal{A}}^* (q_0 a_3[\gamma_{n,n}\mathrm{B}_2[{a_1}^n]]q_0) \\
&\to_{\mathcal{A}}^* (q_0 \mathrm{A}_3[\varepsilon]q_0) \\
&\to_{\mathcal{A}}^* \alpha^{f(n)}
\end{aligned}$$

Seconde étape : En procédant comme dans la preuve du lemme 11.5.11, nous obtenons un automate $\mathcal{A}_1 \in k\text{-}\mathsf{ACD}^{\vec{N}}$ tel que pour tout $n \geq 0$, $m \in [1, n]$

$$(q_0 b_3[\gamma'_{m,n}]q_0) \to_{\mathcal{A}}^* (q_0 \mathrm{B}_3[\varepsilon]q_0)^{g(n-m)}$$

(c'est la règle (C2) de la preuve du lemme 11.5.11) et vérifiant (**Q3**) : il n'y a pas de transitions débutant par $(q_0, \varepsilon, \mathrm{B}_3 w)$.
En ajoutant le nouveau symbole B_2 et en transformant les transitions de façon à traiter toute pile bloquante comme une pile vide, nous obtenons un automate vérifiant (C2). Cette transformation n'introduit pas de nouvelles transitions débutant par $(q_0, \varepsilon, \mathrm{B}_3 w)$ et vérifie donc encore (**Q3**). Les autres dérivations sont obtenues de façon évidente, principalement comme dans la preuve de la proposition 11.5.11.
□

Remarque 11.5.14. *Regardons la suite g comme une série formelle*

$$g = \sum_{n=0}^{\infty} g(n) X^n.$$

La proposition 11.5.12 affirme que la série $\frac{1}{1-Xg}$ appartient à $\mathbb{S}_{k+1}$. En d'autres mots, l'inverse de convolution de toute série formelle de la forme $1 - Xg$, où $g \in \mathbb{S}_k$, appartient à $\mathbb{S}_{k+1}$.

Proposition 11.5.15 (Composition de séries). *Soit $g \in \mathbb{S}_k$, $k \geq 2$, $g(0) = 0$ et $f \in \mathbb{S}_{k+1}^{\vec{N}}$, alors la suite $(f \bullet g)(X) = f(g(X))$ appartient à $\mathbb{S}_{k+1}^{\vec{N}}$.*

Preuve : Pour $n \geq 0$, $(f \bullet g)(n) = f(n) \cdot h(n)$ où

$$h(n) = \sum_{(i_1,\ldots,i_m)\in I_n} g(i_1)\cdots g(i_m),\ I_n = \{(i_1,\ldots,i_m) \mid m \in [1,n],\ i_1,\ldots,i_m \geq 1,\ \sum_{j\in[1,m]} i_j = n\}$$

En utilisant la stabilité par produit établie dans la proposition 11.5.5, il suffit de prouver que la suite $h(n)$ appartient à $\mathbb{S}_{k+1}$. Pour cela, il faut être capable d'énumérer tous les m-uplets de I_n. Codons les éléments de I_n par des mots de $\{a_2, b_2\}^{n-1} \cdot b_2$.

Tout $\vec{p} = (p_1 + 1, \ldots, p_m + 1)$, $p_i \geq 0$, appartenant à I_n est associé de façon bijective au mot $u_{\vec{p}} = a_2{a_1}^{p_1} b_2 a_2 {a_1}^{p_2} b_2 \cdots a_2 {a_1}^{p_m} b_2$ de $\{a_2, b_2\}^{n-1} \cdot b_2$.

Nous pouvons alors représenter chaque élément de I_n par une 2-pile en utilisant le codage des mots de $\{a_2, b_2\}^*$ suivant :

$\forall u = \beta_1 \cdots \beta_n \in \{a_2, b_2\}^n$,

$$\gamma_u = \beta_1[a_1]\beta_2[a_1a_1]\cdots\beta_n[{a_1}^n]c_2[{a_1}^n].$$

Il nous faut maintenant choisir un ordre pour énumérer les m-uplets de I_n. Nous choisissons d'utiliser l'ordre lexical inversé : pour tous $u, v \in \{a_2, b_2\}^n$, nous notons $u <_n v$ ssi $u = u'a_2w$ et $v = v'b_2w$. Nous définissons ainsi un ordre total sur les éléments de $\{a_2, b_2\}^{n-1} \cdot b_2$. Le plus petit élément est $a_2{a_1}^{n-1}b_2$ et le plus grand élément est ${b_2}^n$. Nous définissons la notion de successeur pour cet ordre : si $u = {b_2}^p a_2 w$, le successeur de u est $succ_{lex}(u) = a_2{a_1}^p b_2 w$.

Première étape : Supposons donné $\mathcal{A} = (Q, \{\alpha\}, (A_1, \ldots, A_k), \Delta, q_0) \in (k+1)$-ACD, avec $A_{k+1} \supseteq \{a_{k+1}, \mathrm{A}_{k+1}, b_{k+1}, c_{k+1}\}$, $A_i \supseteq \{a_i\}$ pour $i \in [k, 3]$, $A_2 \supseteq \{a_2, b_2, c_2\}$ et $A_1 = \{a_1\}$, permettant les dérivations de base suivantes :

Initialisation, (I0) : $\forall n \geq 0$,

$$(q_0 a_{k+1}[T_{k,3}[c_2[{a_1}^n]]]q_0) \rightarrow_{\mathcal{A}}^* (q_0 c_{k+1}[T_{k,3}[\gamma_{{a_2}^{n-1}b_2}]]]q_0)$$

Génération des m-uplets, (G1) : pour tout $u \in \{a_2, b_2\}^{n-1} \cdot b_2$, $u \neq {b_2}^n$,

$$(q_0 c_{k+1}[T_{k,3}[\gamma_u]]q_0) \rightarrow_{\mathcal{A}}^* (q_0 c_{k+1}[T_{k,3}[\gamma_{suiv_n(u)}]]q_0)(q_0 b_{k+1}[T_{k,3}[\gamma_u]]q_0)$$

Génération des m-uplets, (G10) :

$$(q_0 c_{k+1}[T_{k,3}[\gamma_{{b_2}^n}]]q_0) \rightarrow_{\mathcal{A}}^* (q_0 b_{k+1}[T_{k,3}[\gamma_{{b_2}^n}]]q_0)$$

Calcul de g, (D2) : pour tous $\ell \geq 1$, $i_0, i_1, \ldots, i_\ell \geq 0$

$$(q_0 b_{k+1}[T_{k,3}[a_2[{a_1}^{i_0}]a_2[{a_1}^{i_1}]\cdots a_2[{a_1}^{i_{\ell-1}}]b_2[{a_1}^{i_\ell}]\Omega_2]]q_0) \rightarrow_{\mathcal{A}}^* (q_0 \mathrm{A}_{k+1}[T_{k,3}[\Omega_2]]q_0)^{g(\ell+1)}$$

Calcul de g, (D20) : pour tout $i \geq 0$,

$$(q_0 b_{k+1}[T_{k,3}[b_2[{a_1}^i]\Omega_2]]q_0) \rightarrow_{\mathcal{A}}^* (q_0 \mathrm{A}_{k+1}[T_{k,3}[\Omega_2]]q_0)^{g(1)}$$

Règles de terminaison, (T0) : pour tout $i \geq 0$

$$(q_0 \mathrm{A}_{k+1}[T_{k,3}[c_2[{a_1}^i]]]q_0) \rightarrow_{\mathcal{A}}^* \alpha.$$

Règles de terminaison, (T1) : pour tout $i \geq 0$

$$(q_0 \mathrm{A}_{k+1}[T_{k,3}[a_2[\Omega_1]\Omega_2]]q_0) \rightarrow^*_{\mathcal{A}} (q_0 b_{k+1}[T_{k,3}[a_2[\Omega_1]\Omega_2]]q_0)$$

Règles de terminaison, (T1') : pour tout $i \geq 0$

$$(q_0 \mathrm{A}_{k+1}[T_{k,3}[b_2[\Omega_1]\Omega_2]]q_0) \rightarrow^*_{\mathcal{A}} (q_0 b_{k+1}[T_{k,3}[b_2[\Omega_1]\Omega_2]]q_0)$$

Vérifions maintenant qu'un automate validant ces dérivations calcule la suite h. Puisque $<_n$ définit un ordre total sur $\{a_2, b_2\}^{n-1}b_2$ dont le plus petit élément est ${a_2}^{n-1}b_2$ et le plus grand est ${b_2}^n$, par la règle d'initialisation (I0), puis itération de (G1), puis (G10), nous obtenons :

$$\begin{array}{rcl} (q_0 a_{k+1}[T_{k,3}[c_2[{a_1}^n]]]q_0) & \rightarrow^*_{\mathcal{A}} & (q_0 c_{k+1}[T_{k,3}[\gamma_{{a_2}^{n-1}b_2}]]]q_0) \\ & \rightarrow^*_{\mathcal{A}} & \prod_{\vec{p}\in I_n} (q_0 b_{k+1}[T_{k,3}[\gamma_{u_{\vec{p}}}]]q_0) \end{array} \quad (37)$$

En partant de chaque facteur codant $\vec{p} = (p_1, \ldots, p_m)$, $p_i \geq 1$ du produit 37, nous obtenons en appliquant itérativement (C2,T1) et (C20,T1) ((C2) lorsque $\gamma_{u_{p_i,\ldots,p_m}}$ commence par a_2 et (C20) lorsque $\gamma_{u_{p_i,\ldots,p_m}}$ commence par b_2) :

$$\begin{array}{rcl} (q_0 b_{k+1}[T_{k,3}[\gamma_{u_{\vec{p}}}]]q_0) & \rightarrow^*_{\mathcal{A}} & (q_0 \mathrm{A}_{k+1}[T_{k,3}[\gamma_{u_{p_2,\ldots,p_m}}]]q_0)^{g(p_1)} \\ & \rightarrow^*_{\mathcal{A}} & (q_0 b_{k+1}[T_{k,3}[\gamma_{u_{p_2,\ldots,p_m}}]]q_0)^{g(p_1)} \\ & \rightarrow^*_{\mathcal{A}} & \cdots \\ & \rightarrow^*_{\mathcal{A}} & (q_0 \mathrm{A}_{k+1}[T_{k,3}[c_2[{a_1}^n]]]q_0)^{g(p_1)\cdots g(p_m)} \end{array} \quad (38)$$

Appliquons à chaque élément du produit (37), la la dérivation (38) associée. Nous obtenons

$$\begin{array}{rcl} (q_0 a_{k+1}[T_{k,3}[c_2[{a_1}^n]]]q_0) & \rightarrow^*_{\mathcal{A}} & \prod_{(p_1,\ldots,p_m)\in I_n} (q_0 \mathrm{A}_{k+1}[T_{k,3}[c_2[{a_1}^n]]]q_0)^{g(p_1)\cdots g(p_m)} \\ & \rightarrow^*_{\mathcal{A}} & \prod_{(p_1,\ldots,p_m)\in I_n} \alpha^{g(p_1)\cdots g(p_m)} \text{ (par règle (T0))} \\ & = & \alpha^{\sum_{(p_1,\ldots,p_m)\in I_n} g(p_1)\cdots g(p_m)} = \alpha^{h(n)} \end{array}$$

L'automate $\mathcal{A}$ calcule donc la suite h.

Seconde étape : Construisons maintenant un automate $\mathcal{A}$ réalisant les dérivations ci-dessus. Par une construction similaire à celle donnée dans la proposition 11.5.12, (et similaire à celle donnée dans la preuve de 11.5.13 si $k = 2$) pour la dérivation (C2), nous obtenons $\mathcal{A}_2 = (Q_2, \emptyset, (B_1, \ldots, B_{k+1}), \Delta_2, q_0) \in (k+1)$-**ACD**, où $B_{k+1} \supseteq \{b_{k+1}, \mathrm{A}_{k+1}\}$, $B_i \supseteq \{a_i\}$ pour $i \in [3, k]$, $B_2 \supseteq \{a_2, b_2, c_2\}$ et $B_1 = \{a_1\}$ et vérifiant les dérivations (C2) et (C20) ainsi que les conditions (**Q2**) et (**Q2**(A_{k+1})) posées dans le lemme 11.5.2.

Définissons $\mathcal{A} = (Q, \{\alpha\}, (A_1, \ldots, A_{k+1}), \Delta, q_0)$ où

$$Q = Q_2 \cup \{q_1, q_2\}, \quad A_i = B_i \text{ pour } i \in [1, k], \quad A_{k+1} = B_{k+1} \cup \{b_{k+1}, a'_{k+1}\},$$

et Δ est l'union de Δ_2 avec les transitions suivantes : pour $w = a_k \cdots a_3$,

(0.1) $\Delta(q_0, \varepsilon, a_{k+1} w c_2 a_1) = (\mathrm{push}_{b_2}, q_1)$

(0.2) $\Delta(q_1, \varepsilon, a_{k+1} w a_2 a_1) = \Delta(q_1, \varepsilon, a_{k+1} w b_2 a_1) = (\mathrm{push}_{a_2}\, \mathrm{pop}_1, q_1)$
(0.3) $\Delta(q_1, \varepsilon, a_{k+1} w a_2) = \Delta(q_1, \varepsilon, a_{k+1} w b_2) = (\mathrm{pop}_2\, \mathrm{change}_{c_{k+1}}, q_0)$

(1.1.1) $\Delta(q_0, \varepsilon, c_{k+1} w b_2 a_1) = (\mathrm{change}_{b_{k+1}}\, \mathrm{push}_{c_{k+1}}, q_2)$
(1.1.2) $\Delta(q_0, \varepsilon, c_{k+1} w a_2 a_1) = (\mathrm{change}_{b_{k+1}}\, \mathrm{push}_{c_{k+1}}\, \mathrm{change}_{b_2}, q_0)$
(1.2.1) $\Delta(q_2, \varepsilon, c_{k+1} w b_2 a_1) = (\mathrm{pop}_2, q_2)$
(1.3.0) $\Delta(q_2, \varepsilon, c_{k+1} w c_2 a_1) = (\mathrm{pop}_{k+1}, q_0)$
(1.3.1) $\Delta(q_2, \varepsilon, c_{k+1} w a_2 a_1) = (\mathrm{change}_{b_2}, q_1)$
(1.4.1) $\Delta(q_1, \varepsilon, c_{k+1} w b_2 a_1) = \Delta(q_1, \varepsilon, c_{k+1} w a_2 a_1) = (\mathrm{pop}_1\, \mathrm{push}_{a_2}, q_1)$
(1.5.1) $\Delta(q_1, \varepsilon, c_{k+1} w a_2) = (\mathrm{pop}_2, q_0)$

(3) $\Delta(q_0, \alpha, \mathrm{A}_{k+1} w c_2 a_1) = (\mathrm{pop}_{k+1}, q_0)$

(4) $\Delta(q_0, \alpha, \mathrm{A}_{k+1} w b_2 a_1) = \Delta(q_0, \alpha, \mathrm{A}_{k+1} w a_2 a_1) = (\mathrm{change}_{b_{k+1}}, q_0)$

Ces transitions sont toutes incompatibles deux à deux. Puisque c_{k+1} et a_{k+1} sont des nouveaux symboles, et que $\mathcal{A}_2$ est déterministe et vérifie la condition ($\mathbf{Q3}(\mathrm{A}_{k+1})$), l'ajout des transitions Δ_1 aux transitions ci-dessus ne modifie pas le déterminisme de Δ. L'automate $\mathcal{A}$ est donc déterministe.

Vérifions maintenant que $\mathcal{A}$ réalise les dérivations décrites dans la première étape.

Les dérivations (C2) et (C20) sont rendues valides par les transitions Δ_2. La règle de terminaison (T0) est obtenue en appliquant la transition (3), les dérivations (T1) et (T1') découlent des transitions (4).

Par les transitions (0.1), puis itération de (0.2), puis (0.3) nous obtenons la dérivation suivante :

$$\begin{array}{rcl}
(q_0 a_{k+1}[T_{k,3}[c_2[{a_1}^n]]]q_0) & \to_{\mathcal{A}} & (q_1 a_{k+1}[T_{k,3}[b_2[{a_1}^n]c_2[{a_1}^n]]]q_0) \\
& \to^*_{\mathcal{A}} & (q_1, a_{k+1}[T_{k,3}[a_2[\varepsilon]a_2[a_1]\cdots a_2[{a_1}^{n-1}]b_2[{a_1}^n]c_2[{a_1}^n]]]q_0) \\
& \to_{\mathcal{A}} & (q_0, c_{k+1}[T_{k,3}[a_2[a_1]\cdots a_2[{a_1}^{n-1}]b_2[{a_1}^n]c_2[{a_1}^n]]]q_0) \\
& = & (q_0 c_{k+1}[T_{k,3}[\gamma_{{a_2}^{n-1}b_2}]]q_0)
\end{array}$$

La dérivation initiale (I0) est donc bien réalisée par l'automate.

Vérifions maintenant que les règles de génération des m-uplets sont valides. Pour montrer (G1), nous distinguons deux cas selon la première lettre de $u \in \{a_2, b_2\}^{n-1} \cdot b_2$, $u \neq {b_2}^n$:
Cas 1 : si $u = {b_2}^i a_2 w$, alors, soit ω la 2-pile telle que $\gamma_u = b_2[a_1]\cdots b_2[{a_1}^i]a_2[{a_1}^{i+1}]\omega$, nous obtenons la dérivation suivante :

$$\begin{array}{rcl}
 & & (q_0 c_{k+1}[T_{k,3}[\gamma_u]]q_0) \\
(\text{par } 1.1.1) & \to_{\mathcal{A}} & (q_2 c_{k+1}[T_{k,3}[\gamma_u]]q_0)(q_0 b_{k+1}[T_{k,3}[\gamma_u]]q_0) \\
(\text{par } (1.2.1)) & \to^i_{\mathcal{A}} & (q_2 c_{k+1}[T_{k,3}[a_2[{a_1}^{i+1}]\omega]]q_0)(q_0 b_{k+1}[T_{k,3}[\gamma_u]]q_0) \\
(\text{par } (1.3.1)) & \to_{\mathcal{A}} & (q_1 c_{k+1}[T_{k,3}[b_2[{a_1}^{i+1}]\omega]]q_0)(q_0 b_{k+1}[T_{k,3}[\gamma_u]]q_0) \\
(\text{par } (1.4.1)) & \to^i_{\mathcal{A}} & (q_1 c_{k+1}[T_{k,3}[a_2[\varepsilon]\cdots a_2[{a_1}^i]b_2[{a_1}^{i+1}]\omega]]q_0)(q_0, b_{k+1}[T_{k,3}[\gamma_u]]q_0) \\
(\text{par } (1.5.1)) & \to_{\mathcal{A}} & (q_0 c_{k+1}[T_{k,3}[\gamma_{suiv_{lex}(u)}]]q_0)(q_0 b_{k+1}[T_{k,3}[\gamma_u]]q_0)
\end{array}$$

Cas 2 : si $u = a_2 w$, alors, soit ω tel que $\gamma_u = a_2[a_1]\omega$.

Nous obtenons la dérivation suivante :

$$\begin{array}{rcl} & & (q_0 c_{k+1}[T_{k,3}[\gamma_u]]q_0) \\ (\text{par (1.1.2)}) & \to_{\mathcal{A}} & (q_0 c_{k+1}[T_{k,3}[b_2[a_1]\omega]]q_0)(q_0 b_{k+1}[T_{k,3}[\gamma_u]]q_0) \\ & = & (q_0 c_{k+1}[T_{k,3}[\gamma_{suiv_{lex}(u)}]]b_{k+1}[T_{k,3}[\gamma_u]]q_0) \end{array}$$

Montrons pour finir que la règle de génération (G10) est valide :

$$\begin{array}{rcl} & & (q_0 c_{k+1}[T_{k,3}[\gamma_{b_2{}^n}]]q_0) \\ (\text{par (1.1.1)}) & \to_{\mathcal{A}} & (q_2 c_{k+1}[T_{k,3}[\gamma_{b_2{}^n}]]q_0)(q_0 b_{k+1}[T_{k,3}[\gamma_{b_2{}^n}]]q_0) \\ (\text{par (1.2.1)}) & \to^n_{\mathcal{A}} & (q_2 c_{k+1}[T_{k,3}[c_2[a_1{}^n]]]q_0)(q_0 b_{k+1}[T_{k,3}[\gamma_{b_2{}^n}]]q_0) \\ (\text{par (1.3.0)}) & \to_{\mathcal{A}} & (q_0 b_{k+1}[T_{k,3}[\gamma_{b_2{}^n}]]q_0) \end{array}$$

Toutes les dérivations utilisées dans la première étape sont valides, la proposition est donc prouvée.

□

Proposition 11.5.16 (Composition de suites). *Soient $k_1, k_2 \geq 1$, $f \in \mathbb{S}_{k_1+1}$ et $g \in \mathbb{S}^{\vec{N}}_{k_2+1}$, alors $g \circ f \in \mathbb{S}^{\vec{N}}_{k_1+k_2+1}$.*

Construction : En utilisant le 11.5.1, et après un choix approprié d'ensembles concrets états et de symboles de piles, nous obtenons un automate $\mathcal{A}_1 \in (k_1+1)\text{-}\mathsf{ACD}^{\vec{N}}(A_1, \ldots, A_{k_1+1})$, ainsi qu'un automate $\mathcal{A}_2 \in (k_2+1)\text{-}\mathsf{ACD}^{\vec{N}}(B_1, \ldots, B_{k_2+1})$ et vérifiant les conditions :
(**P1.1**) $\forall \Omega_k \in \mathcal{I}_k, (q_0, a_{k_1+1}[a_{k_1}[\cdots[a_2[a_1{}^n]]\cdots]], q_0) \to^*_{\mathcal{A}_1} (q_0, \mathrm{A}_{k_1+1}[\varepsilon], q_0)^{f(n)}$.
(**P1.2**) $\forall \Omega_k \in \mathcal{I}_k, (r_0, b_{k_2+1}[b_{k_2}[\cdots[b_2[b_1{}^n]]\cdots]], r_0) \to^*_{\mathcal{A}_2} (r_0, \mathrm{B}_{k_2+1}[\varepsilon], r_0)^{g(n)}$.
(**P2.1**) Δ_1 n'admet pas de membres gauches de la forme $(q_0, \varepsilon, \varepsilon)$, $q \in Q_1$.
(**P2.2**) Δ_2 n'admet pas de membres gauches de la forme $(q, \varepsilon, \varepsilon)$, $q \in Q_2$.
(**P3.1**) Δ_1 n'admet pas de membres gauches de la forme $(q, \varepsilon, \mathrm{A}_{k+1} \cdot w)$.
(**P3.2**) Δ_2 n'admet pas de membres gauches de la forme $(r_0, \varepsilon, \mathrm{B}_{k+1} \cdot w, \vec{o})$.
(**P4**) $A_1 \cap B_{k_2+1} = \{a_1\} = \{\mathrm{B}_{k_2+1}\}$.
Nous considérons l'automate (k_1+k_2+1)-ACD $\mathcal{A} = (\vec{N}, Q, \{\alpha\}, (C_1, \ldots, C_{k_1+k_2+1}), \Delta, (q_0, r_0)) \in (k_1+k_2+1)\text{-}\mathsf{ACD}$ où : $Q = Q_1 \times Q_2$, et $C_{k_2+k_1+1} = A_{k_1+1}, \ldots, C_{k_2+2} = A_2$, $C_{k_2+1} = B_{k_2+1}, \ldots, C_1 = B_1$, et Δ est l'union des transitions suivantes.
Transitions héritées de $\mathcal{A}_1$:

pour tout $\Delta_1(q_1, \varepsilon, w_1) = (\text{instr}, p_1)$, $w_1 \in \text{top}((k_1+1)\text{-}\mathsf{NPile}$, pour tout $\vec{o} \in \{0,1\}^{|\vec{N}|}$,
(1) $\Delta((q_1, r_0), \varepsilon, w_1, \vec{o}) = (\text{instr}^{+k_2}, (p_1, r_0))$,

où la notation instr^{+k} signifie :
- si instr $= \text{pop}_i$ alors $\text{instr}^{+k} = \text{pop}_{i+k}$
- si instr $= \text{push}_i(a)$ alors $\text{instr}^{+k} = \text{push}_{i+k}(a)$
- si instr $= \text{change}_i(a)$ alors $\text{instr}^{+k} = \text{change}_{i+k}(a)$.

Transitions héritées de $\mathcal{A}_2$:

pour tout $\Delta_2(r, \varepsilon, w_2, \vec{o}) = (\text{instr}, r')$, $w_2 \in \text{top}((k_2+1)\text{-}\mathsf{NPile}$, $r, r' \in Q_2$, et pour tous $q_1 \in Q_1$, $w_1 \in A_{k_1+1} \cdots A_2$
(2) $\Delta((q_1, r), \varepsilon, w_1 \cdot w_2, \vec{o}) = (\text{instr}, (q_1, r'))$.

Preuve : Prouvons que l'automate ci-dessus $\mathcal{A}$ est déterministe.
Le fait que les automates initiaux $\mathcal{A}_i$ $(i = 1, 2)$ soient déterministes assure que deux de transitions distinctes du même type (i) sont incompatibles. Supposons maintenant qu'il existe une transition de type (1) compatible avec une transition de type (2). Nous avons alors

$$((q, r_0), w_1) = ((q_1, r), w_1' \cdot w_2')$$

pour $q, q_1 \in Q_1$, $r \in Q_2$, $w_1 \in \mathrm{top}((k_1+1)\text{-}\mathsf{NPile}(A_1, \ldots, A_{k_1+1}))$, $w_1' \in A_{k_1+1} \cdots A_1$, $w_2' \in \mathrm{top}((k_2+1)\text{-}\mathsf{NPile})(B_1, \ldots, B_{k_2+1})$. La seule possibilité pour une telle égalité est

$$r = r_0 \ \text{ et } \ [w_2' = a_1 = \mathrm{B}_{k_2+1} \text{ ou } w_2' = \varepsilon]$$

Mais, par la condition (**P3.2**), il n'y a aucune transition de Δ_2 démarrant avec $(r_0, \mathrm{B}_{k_2+1})$, de même, par (**P2.2′**), il n'y a aucune transition de Δ_2 démarrant avec (r_0, ε). Finalement, nous sommes sûrs que $\mathcal{A}$ est un (k_1+k_2+1)-ACD déterministe avec compteur b_1.
Vérifions maintenant que

$$((q_0, r_0)a_{k_1+1}[\cdots a_2[b_{k_2+1}[\cdots[b_2[{b_1}^n]]\cdots]]\cdots](q_0, r_0)) \to_{\mathcal{A}}^* ((q_0, r_0)\mathrm{A}_{k_1+1}[\varepsilon](q_0, r_0))^{f(g(n))} \tag{39}$$

Nous définissons la fonction partielle $\varphi : (k_2+1)\text{-}\mathsf{NPile}(B_1, \ldots, B_{k_2+1}) \to 1\text{-}\mathsf{NPile}(A_1)$ vérifiant pour tout $\omega \in (k_2+1)\text{-}\mathsf{NPile}(B_1, \ldots, B_{k_2+1})$

$$\varphi(\omega) = {a_1}^n \Leftrightarrow (r_0 \omega r_0) \to_{\mathcal{A}_2}^* (r_0 \mathrm{B}_{k_2+1} r_0)^n \text{ ou bien } \omega = \varepsilon \text{ et } \varphi(\omega) = \varepsilon \tag{40}$$

ainsi, $\varphi(\omega)$ est définie exactement pour chaque ω tel que $(r_0 \omega r_0)$ se dérive (modulo $\mathcal{A}_2$) en $(r_0 \mathrm{B}_{k_2+1} r_0)^*$.

Lemme 11.5.17. *Pour tout (k_1+1)-terme $T_1[\Omega_1]$, où Ω_1 indéterminé de niveau 1 admettant une et une seule occurrence dans T_1, pour tous $\omega_1 \in \mathrm{Dom}(\varphi)$ et $\omega \in (k_1+1)\text{-}\mathsf{NPile}$,*

$$\textit{si } (q, \varepsilon, T_1[\varphi(\omega_1)]) \vdash_{\mathcal{A}_1} (q', \varepsilon, \omega)$$

alors, il existe $\omega_2 \in \mathrm{Dom}(\varphi)$ et un (k_1+1)-terme $T_2[\Omega_1]$, tels que pour tout $p \in Q$

$$((q, r_0){T_1}^{+k_2}[\omega_1](p, r_0)) \to_{\mathcal{A}}^* ((q', r_0){T_2}^{+k_2}[\omega_2](p, r_0)) \textit{ et } T_2[\varphi(\omega_2)] = \omega$$

Preuve : Supposons que la transition appliquée est $(q, \varepsilon, w, \mathrm{instr}, q')$, c'est à dire

$$(q, \varepsilon, T_1[\varphi(\omega_1)]) \vdash_{\mathcal{A}_1} (q', \varepsilon, \mathrm{instr}(T_1[\varphi(\omega_1)])) \text{ et } \omega = \mathrm{instr}(T_1[\varphi(\omega_1)]) \tag{41}$$

Nous distinguons deux cas selon la position de l'indéterminée dans T_1 :
Cas 1 : si Ω_1 n'apparaît pas dans les symboles de tête de T_1, alors $\mathrm{instr}(T_1)$ est défini. Posons $T_2[\Omega_1] = \mathrm{instr}(T_1[\Omega_1])$. Nous avons alors $\omega = T_2[\varphi(\omega_1)]$.

Puisque $\mathrm{top}_1(T) \neq \Omega_1$, $|\mathrm{top}({T_1}^{+k_2}[\omega])| \leq k_1 + 1$, et en utilisant la transition de type (1) associée à la transition appliquée dans (41) (et ayant pour test $\chi_{\vec{N}}(0)$), nous obtenons

$$((q, r_0){T_1}^{+k_2}[\omega_1](p, r_0)) \to_{\mathcal{A}} ((q', r_0){T_2}^{+k_2}[\omega_1](p, r_0))$$

Cas 2 : si l'occurrence de Ω_1 apparaît en tête de T_1, nous distinguons deux sous-cas selon le niveau de l'instruction instr.

Cas 2.1 si instr est de niveau supérieur à 1, alors $\mathrm{instr}(T_1)$ est défini. Posons $T_2 = \mathrm{instr}(T_1)$, dans ce cas $\mathrm{instr}^{+k_2}({T_1}^{+k_2}) = {T_2}^{+k_2}$ et $\omega = T_2[\varphi(\omega_1)]$. Nous distinguons à nouveau deux sous-cas selon la valeur de $\varphi(\omega_1)$:
Cas 2.1.1 : si $\varphi(\omega_1) = \varepsilon$, alors, en appliquant les transitions de type (2), puis la transition de type (1) associée au calcul (41) :

$$\begin{array}{lll}((q,r_0){T_1}^{+k_2}[\omega_1](p,r_0)) & \to^*_{\mathcal{A}} & ((q,r_0){T_1}^{+k_2}[\varepsilon](p,r_0)) \\ & \to^*_{\mathcal{A}} & ((q',r_0)\,\mathrm{instr}^{+k_2}({T_1}^{+k_2})[\varepsilon](p,r_0))\end{array}$$

Le lemme est donc vérifié pour $\omega_2 = \varepsilon$ puisque $T_2[\varphi(\omega_2)] = T_2[\varepsilon] = \omega$.
Cas 2.1.2 : si $\varphi(\omega_1) = {a_1}^{n+1}$, $n \geq 0$, alors, par définition de φ, il existe $\hat{\omega}_1 \in \mathrm{Dom}(\varphi)$ tel que :

$$(r_0\omega_1 r_0) \to^*_{\mathcal{A}_2} (r_0 \mathrm{B}_{k_2+1} r_0)(r_0\hat{\omega}_1 r_0) \text{ et } \varphi(\hat{\omega}_1) = {a_1}^n$$

En appliquant les transitions de type (2), puis la transition de type (1) associée au calcul (41) :

$$\begin{array}{lll}(q,r_0){T_1}^{+k_2}[\omega_1](p,r_0)) & \to^*_{\mathcal{A}_1} & ((q,r_0){T_1}^{+k_2}[\mathrm{B}_{k_2+1}\hat{\omega}_1](p,r_0)) \\ & \to^*_{\mathcal{A}_1} & ((q',r_0)\,\mathrm{instr}^{+k_2}({T_1}^{+k_2}[\mathrm{B}_{k_2+1}\hat{\omega}_1])(p,r_0))\end{array}$$

Le lemme est donc vérifié pour $\omega_2 = \mathrm{B}_{k_2+1}\hat{\omega}_1$ puisque $T_2[\varphi(\omega_2)] = T_2[{a_1}^{n+1}] = \omega$.

Cas 2.2 si instr est de niveau 1, nous posons $T_2 = T_1$. Nous avons alors $\omega = T_2[\mathrm{instr}(\varphi(\omega_1))]$. Nous distinguons encore deux cas selon la valeur de $\varphi(\omega_1)$:
Cas 2.2.1 : si $\varphi(\omega_1) = \varepsilon$, alors, en appliquant les transitions de type (2), puis la transition de type (1) associée au calcul (41) :

$$\begin{array}{lll}(q,r_0){T_1}^{+k_2}[\omega_1](p,r_0)) & \to^*_{\mathcal{A}_1} & ((q,r_0){T_1}^{+k_2}[\varepsilon](p,r_0)) \\ & \to^*_{\mathcal{A}_1} & ((q',r_0){T_1}^{+k_2}[\mathrm{instr}^{+k_2}(\varepsilon)](p,r_0))\end{array}$$

(dans ce cas $\mathrm{instr} = \mathrm{push}_{a_1}$). Le lemme est donc vérifié pour $\omega_2 = \mathrm{instr}^{+k_2}(\varepsilon)$ puisque $T_2[\varphi(\omega_2)] = T_2[\varphi(\varepsilon)] = \omega$.
Cas 2.2.2 : si $\varphi(\omega) = {a_1}^{n+1}$, $n \geq 0$, alors, en appliquant les transitions de type (2), puis la transition de type (1) associée au calcul (41) :

$$\begin{array}{lll}(q,r_0){T_1}^{+k_2}[\omega_1](p,r_0)) & \to^*_{\mathcal{A}_1} & ((q,r_0){T_1}^{+k_2}[\mathrm{B}_{k_2+1}\hat{\omega}_1](p,r_0)) \\ & \to^*_{\mathcal{A}_1} & ((q',r_0){T_1}^{+k_2}[\mathrm{instr}^{+k_2}(\mathrm{B}_{k_2+1}\hat{\omega}_1)](p,r_0))\end{array}$$

Le lemme est donc vérifié pour $\omega_2 = \mathrm{instr}^{+k_2}(\mathrm{B}_{k_2+1}\hat{\omega}_1)$ puisque

$$T_2[\varphi(\omega_2)] = T_2[\mathrm{instr}({a_1}^{n+1})] = \omega.$$

□

Nous étendons maintenant ce lemme aux dérivations en introduisant une application partielle $\Phi : V_{\mathcal{A}} \to V_{\mathcal{A}_1}$, de l'ensemble des variables de $\mathcal{A}$ (défini dans §11.3 par l'équation (16)) dans l'ensemble des variables de $\mathcal{A}_1$. Pour tout $T[\Omega_1,\ldots,\Omega_n] \in (k_1+1)\text{-}\mathsf{Terme}(A_1,\ldots,A_{k_1+1})$, Ω_i indéterminées de niveau 1 et $\omega_1,\ldots,\omega_n \in (k_2+1)\text{-}\mathsf{NPile}(B_1,\ldots,B_{k_2+1})$, $q \in Q_1$, nous définissons

$$\Phi((q,r_0)T^{+k_2}[\omega_1,\ldots,\omega_n](q',r_0)) = (qT[\varphi(\omega_1),\ldots,\varphi(\omega_n)]q')$$

ainsi $\Phi(V)$ est défini exactement pour les variables $V = ((q, r_0)T^{+k_2}[\omega_1, \ldots, \omega_n](q', r_0))$ telles que pour tout Ω_i apparaissant dans T, $\omega_i \in \mathrm{Dom}(\varphi)$. Nous étendons la fonction Φ aux mots en posant :

$$\Phi(V_1 V_2 \cdots V_m) = \Phi(V_1)\Phi(V_2)\cdots\Phi(V_m) \text{ si pour tout } i \ , V_i \in \mathrm{Dom}(\Phi)$$

et $\Phi(V_1 V_2 \cdots V_m)$ est indéfini sinon.

Lemme 11.5.18. *Si $U \in \mathrm{Dom}(\Phi)$ et $U_1' \in {V_{\mathcal{A}_1}}^*$ sont tels que*

$$\Phi(U) \to_{\mathcal{A}_1} U_1'$$

alors, il existe un mot $U' \in \mathrm{Dom}(\Phi)$ tel que

$$U \to_{\mathcal{A}}^* U' \ \ \& \ \ \Phi(U') = U_1'.$$

Preuve : Il suffit de le vérifier dans le cas où U est réduit à une variable. Supposons $U = (q, r_0)T[\omega_1, \omega_2, \ldots, \omega_n](q', r_0)$ où $T[\Omega_1, \Omega_2, \ldots, \Omega_n]$ est un (k_1+1)-terme, tous les ω_i appartiennent à $\mathrm{Dom}(\varphi)$ et $q, q' \in Q_1$. Sans perte de généralité, nous pouvons supposer que chaque Ω_i admet une unique occurrence dans T et pour tous $1 \leq i < j \leq n$, Ω_i apparaît à gauche de Ω_j. Nous supposons que

$$\Phi(U) \to_{\mathcal{A}_1} U_1'. \tag{42}$$

Nous distinguons deux cas, dépendants du type de règle utilisé dans la dérivation (42).
Cas 1 : Règle de décomposition.
Cela signifie que $T = T' \cdot T''$ et donc, il existe $n' \in [1, n]$ tel que

$$\begin{array}{rcl} qT[\varphi(\omega_1), \ldots, \varphi(\omega_n)]q' & \to_{\mathcal{A}_1} & qT_1[\varphi(\omega_1), \ldots, \varphi(\omega_{n'})]q'' \cdot q''T_2[\varphi(\omega_{n'+1}), \ldots, \varphi(\omega_n)]q' \\ & = & U_1'. \end{array}$$

Dans ce cas, par une règle de décomposition, $U \to_{\mathcal{A}}^* U'$ avec

$$U' = (q, r_0){T_1}^{+k_2}[\omega_1, \ldots, \omega_{n'}](q'', r_0) \cdot (q'', r_0){T_2}^{+k_2}[\omega_{n'+1}, \ldots, \omega_n](q', r_0)$$

vérifie la conclusion du lemme.
Cas 2 : Règle de transition.
Il existe $p \in Q_1$ et une instruction instr tels que (42) se reformule :

$$(q, T[\varphi(\omega_1), \ldots, \varphi(\omega_n)], q') \to_{\mathcal{A}_1} (p, \mathrm{instr}(T[\varphi(\omega_1), \ldots, \varphi(\omega_n)]), q')$$

En appliquant le lemme précédent, il existe $U' \in \mathrm{Dom}(\Phi)$ tel que $U \to_{\mathcal{A}_1}^* U'$ et $\Phi(U') = (p, T[\varphi(\omega_1), \ldots, \varphi(\omega_n)], q')$.

□

Nous pouvons maintenant prouver la dérivation (39). Posons $k = k_1 + k_2 + 1$, nous remarquons que,

$$\begin{array}{rcl} \Phi((q_0, r_0){T_{k,k_2+1}}^{+k_2}[b_{k_2}[\cdots b_2[{b_1}^n]\cdots]](q_0, r_0)) & = & (q_0 T_{k,k_2+1}[{a_1}^{g(n)}]q_0) \\ & \to_{\mathcal{A}_1}^* & (q_0 \mathrm{A}_{k_1+1}[\varepsilon]q_0)^{f(g(n))} \end{array}$$

En appliquant itérativement le lemme 11.5.18, nous obtenons un $U' \in \mathrm{Dom}(\Phi)$ tel que :

$$((q_0, r_0)T_{k,k_2+1}{}^{+k_2}[b_{k_2}[\cdots b_2[b_1{}^n]\cdots]](q_0, r_0)) \to^*_{\mathcal{A}} U' \quad \text{et} \quad \Phi(U') = (q_0 \mathrm{A}_{k_1+1}[\varepsilon]q_0)^{f(g(n))}.$$

Mais la seule valeur possible pour une pré-image par Φ de $(q_0 \mathrm{A}_{k_1+1}[\varepsilon]q_0)^{f(g(n))}$ est

$$U' = ((q_0, r_0)\mathrm{A}_{k_1+1}[\varepsilon](q_0, r_0))^{f(g(n))},$$

car $\mathrm{A}_{k+1}[\varepsilon]$ ne peut correspondre à aucun terme sans indéterminé de niveau 1. La proposition 11.5.16 est donc prouvée.

□

Nous résumons dans le théorème suivant les propriétés de clôture démontrées dans la section 11.5.

Théorème 11.5.19.

0- Pour tout $f \in \mathbb{S}^{\vec{N}}_{k+1}$, $k \geq 1$, et tout entier $c \in \mathbb{N}$,
les suites Ef (le décalage de f), $f + \frac{c}{1-X}$, appartiennent à $\mathbb{S}^{\vec{N}}_{k+1}$;
si $\forall n \in \mathbb{N}, f(n) \geq c$ alors $f - \frac{c}{1-X}$ appartient à $\mathbb{S}^{\vec{N}}_{k+1}$;
la suite définie par $0 \mapsto c, n+1 \mapsto f(n)$ appartient à $\mathbb{S}^{\vec{N}}_{k+1}$.
1- Pour toutes suites $f, g \in \mathbb{S}^{\vec{N}}_{k+1}$, avec $k \geq 1$, la suite $f + g$ appartient à $\mathbb{S}^{\vec{N}}_{k+1}$.
2- Pour toutes suites $f, g \in \mathbb{S}^{\vec{N}}_{k+1}$, avec $k \geq 2$, la suite $f \odot g$ (le produit ordinaire), appartient à $\mathbb{S}^{\vec{N}}_{k+1}$ et pour tout $f' \in \mathbb{S}^{\vec{N}}_{k+2}$, f'^g appartient à $\mathbb{S}^{\vec{N}}_{k+2}$.
3- Pour toutes suites $f \in {\mathbb{S}_{k+1}}^{\vec{N}}$ et $g \in \mathbb{S}_k$, pour $k \geq 2$, les suites $f \times g$ (le produit de convolution) et $f \bullet g$ (la substitution de série) appartiennent à $\mathbb{S}^{\vec{N}}_{k+1}$.
4- Pour toute suite $g \in \mathbb{S}_k$, avec $k \geq 2$, la suite f définie par : $f(0) = 1$ et $f(n+1) = \sum_{m=0}^{n} f(m) \cdot g(n-m)$ (l'inverse de convolution de $1 - X \times f$) appartient à $\mathbb{S}_{k+1}$.
5- Pour toutes suites $f \in \mathbb{S}_k$ et $g \in \mathbb{S}^{\vec{N}}_{\ell}$, pour $k, l \geq 2$, la suite $f \circ g$ (la suite composition) appartient à $\mathbb{S}^{\vec{N}}_{k+\ell-1}$.
6- Pour tout $k \geq 2$ et pour tout système d'équations récurrente exprimées par des polynômes dans $\mathbb{S}^{\vec{N}}_{k+1}[X_1, \ldots, X_p]$, avec conditions initiales dans $\mathbb{N}$, toute solution appartient à $\mathbb{S}^{\vec{N}}_{k+1}$.
7- pour tout $k \geq 2$ et pour tout système d'équations récurrentes exprimées par des polynômes à indéterminées $X_1, \ldots, X_p$, coefficients dans $\mathbb{S}^{\vec{N}}_{k+2}$, exposants dans $\mathbb{S}^{\vec{N}}_{k+1}$ et conditions initiales dans $\mathbb{N}$, toute solution appartient à $\mathbb{S}^{\vec{N}}_{k+2}$.

Preuve : Le point (0) est évident. Tous les autres points sont prouvés dans les propositions précédentes.

□

Chapitre 12

Suites doubles d'entiers

Nous généralisons maintenant la notion de suite k-calculable au cas des suites à deux indices $(s(m,n))_{m,n\geq 0}$ qui sera nécessaire à l'étude des suites simples de nombres rationnels (voir chapitre 14). Cette étude n'est pas exhaustive, nous ne traitons que les cas qui nous utiles par la suite.

Définition 12.0.20. *Soit $k \geq 3$. La suite double $f(m,n)$, $0 \leq m \leq n$ est appelée suite double* $(k, \vec{N})$-calculable *(ou k-calculable lorsque $\vec{N} = \vec{\emptyset}$) ssi, il existe $\mathcal{A} \in k\text{-}\mathsf{ACD}^{\vec{N}}$ tel que, pour tous $0 \leq m \leq n$,*

$$(q_0, \alpha^{f(m,n)}, a_k[\dots a_3[\gamma_{m,n}]\dots]) \vdash_{\mathcal{A}}^{*} (q_0, \varepsilon, \varepsilon),$$

où $\gamma_{m,n}$ désigne la 2-pile $a_2[{a_1}^m]\cdots a_2[{a_1}^{n-1}]b_2[{a_1}^n]$.

Nous notons $\mathbb{S}_k^{(2),\vec{N}}$ l'ensemble des suite doubles $(k, \vec{N})$-calculables.

Remarque 12.0.21. *Contrairement au codage des couples d'indice (m,n) utilisé dans les preuves des propositions 11.5.10 et 11.5.12 traitant des produits de convolution pour les suites simples, le couple (m,n) apparaît au niveau 1 de la pile et les contrôles pourront donc porter sur m et sur n (lorsque $m = n$). La contrepartie est que les compteurs ne peuvent pas évoluer simultanément, et n ne pourra être modifié que lorsque $m = n$.*

Lemme 12.0.22. *Soit $k \geq 2$. Soient $t_{1,1}, t_{1,2}, t_{2,1}, t_{2,2}, s_1, s_2 \in \mathbb{S}_{k+1}{}^{\vec{N}}$. Considérons les suites doubles f_1, f_2 définies par : pour tout $0 \leq \ell < n$,*

$$\begin{pmatrix} f_1(\ell,n) \\ f_2(\ell,n) \end{pmatrix} = \begin{pmatrix} t_{1,1}(\ell) & t_{1,2}(\ell) \\ t_{2,1}(\ell) & t_{2,2}(\ell) \end{pmatrix} \cdot \begin{pmatrix} f_1(\ell+1,n) \\ f_2(\ell+1,n) \end{pmatrix}$$

et

$$f_1(n,n) = s_1(n), \quad f_2(n,n) = s_2(n).$$

Alors $f_1, f_2 \in \mathbb{S}_{k+1}^{(2),\vec{N}}$.

Preuve : Nous utilisons les notations :

$$\gamma_{m,n} = a_2[{a_1}^m]\cdots a_2[{a_1}^{n-1}]b_2[{a_1}^n], T_{i,j+1}[\Omega_j] = a_i[\cdots[a_{j+1}[\Omega_j]]\cdots]$$

Première étape :
Supposons donné un automate $\mathcal{A} = (Q, \{\alpha\}, (A_1, \dots, A_{k+1}), \vec{N}, \Delta, q_0) \in (k+1)\text{-}\mathsf{ACD}$ tel que A_{k+1} contient au moins les symboles : $b_{k+1,1}, b_{k+1,2}$ (symboles pour f_1, f_2), $d_{k+1,1}, \mathrm{D}_{k+1,1}$, et

$d_{k+1,2}, \mathrm{D}_{k+1,2}$, (symboles pour s_1, s_2) et $c_{k+1,i,i}, \mathrm{C}_{k+1,i,j}$, pour $i,j \in \{1,2\}$ (symboles pour $t_{i,j}$).
Supposons que les règles suivantes sont valides :
Règle d'initialisation, (D0) : pour $i \in \{1,2\}$

$$(q_0 a_{k+1,i}[T_{k,3}[\gamma_{n,n}]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 b_{k+1,i}[T_{k,2}[{a_1}^n]\Omega_k]q_0)$$

Règles des coefficients, (C1) : $i,j \in \{1,2\}$

$$(q_0 c_{k+1,i,j}[T_{k,3}[b_2[{a_1}^n]]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 \mathrm{C}_{k+1,i,j}[\Omega_k]q_0)^{t_{i,j}(n)}$$

Règles des suites simples, (S2) : pour $i \in \{1,2\}$

$$(q_0 b_{k+1,i}[T_{k,2}[{a_1}^n]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 \mathrm{A}_{k+1,i}[\Omega_k]q_0)^{s_i(n)}$$

Règles de recollement, (Ri(i,j)) : pour $0 \leq m < n$

$$(q_0 a_{k+1,i}[T_{k,3}[\gamma_{m,n}]\Omega_k]q_0) \rightarrow^*_{\mathcal{A}} \prod_{j\in\{1,2\}} (q_0 c_{k+1,i,j}[T_{k,3}[b_2[{a_1}^n]]T_{k,3}[\gamma_{m+1,n}]\Omega_k]q_0)$$

Règles de recollement, (R(i,j)j) : $\forall \omega_k \in k\text{-Pile}$

$$(q_0 \mathrm{C}_{k+1,i,j}[\omega_k]q_0) \rightarrow^*_{\mathcal{A}} (q_0 a_{k+1,j}[\omega_k]q_0)$$

Règle de terminaison, (T3) :

$$(q_0 \mathrm{A}_{k+1,i}[\varepsilon]q_0) \rightarrow^*_{\mathcal{A}} \alpha$$

Prouvons par induction descendante sur m, que, pour tout $0 \leq m \leq n$, la propriété $\mathbf{P}(m,n)$ suivante est vraie :

$$(q_0 a_{k+1,i}[T_{k,3}[\gamma_{m,n}]]q_0) \rightarrow^*_{\mathcal{A}} \alpha^{f_i(m,n)}$$

Base : $m = n$.
Nous obtenons la dérivation suivante :

$$\begin{aligned}
(q_0 a_{k+1,i}[T_{k,3}[\gamma_{n,n}]]q_0) &\rightarrow^*_{\mathcal{A}} (q_0 b_{k+1,i}[T_{k,2}[{a_1}^n]]q_0) \text{ (par règle (I0))} \\
&\rightarrow^*_{\mathcal{A}} (q_0 \mathrm{A}_{k+1,i}[\varepsilon]q_0)^{f_i(n,n)} \text{ (par (S2))} \\
&\rightarrow^*_{\mathcal{A}} \alpha^{f_i(n,n)} \text{ (par règle (T3))}
\end{aligned}$$

ce qui établit $\mathbf{P}(n,n)$.
Pas d'induction : Supposons $\mathbf{P}(m+1,n)$ pour $0 \leq m < n$. Nous obtenons la dérivation suivante :

$$\begin{aligned}
(q_0 a_{k+1,i}[T_{k,3}[\gamma_{m,n}]]q_0) &\rightarrow^*_{\mathcal{A}} \prod_{j\in\{1,2\}} (q_0 c_{k+1,i,j}[T_{k,3}[b_2[{a_1}^n]]T_{k,3}[\gamma_{m+1,n}]]q_0) \text{ (par (Ri(i,j)))} \\
&\rightarrow^*_{\mathcal{A}} \prod_{j\in\{1,2\}} (q_0 \mathrm{C}_{k+1,i,j}[T_{k,3}[\gamma_{m+1,n}]]q_0)^{t_{i,j}(n)} \text{ (par (C1))} \\
&\rightarrow^*_{\mathcal{A}} \prod_{j\in\{1,2\}} (q_0 a_{k+1,2}[T_{k,3}[\gamma_{m+1,n}]]q_0)^{t_{i,j}(n)} \text{ (par (R(i,j)j))} \\
&\rightarrow^*_{\mathcal{A}} (\alpha^{f_1(m+1,n)})^{t_{i,1}(n)} (\alpha^{f_2(m+1,n)})^{t_{i,2}(n)} \text{ (par } \mathbf{P}(m+1,n))
\end{aligned}$$

ce qui établi $\mathbf{P}(m,n)$.
Seconde étape :
D'après le lemme 11.5.2 il existe des automate $\mathcal{B}_{i,j}$ (resp. $\mathcal{C}_i$) calculant les suites $t_{i,j}$ (resp. s_i), et vérifiant les conditions ($\mathbf{Q1}, \mathbf{Q2}, \mathbf{Q3}$). Ces automates fournissent les ensembles de transitions permettant (C1) et (S2). Comme usuellement, nous choisissons les ensembles concrets d'états et d'alphabets de piles, de façon à ce que le seul état commun des automates soit q_0 et les seuls symboles de piles communs soient $\{a_1, a_2, a_3, \ldots, a_k, \mathrm{A}_k\}$. Chacune des règles (R(i,j)j) et (T3) peut être facilement réduite en une unique transition. Les règles (I0) et (Ri(i,j)) peuvent être décomposées en un nombre fini de transitions, avec états intermédiaire distincts de tous les états déjà utilisés. les variables de départ des règles étant disjoints, les transitions peuvent être choisies incompatibles deux à deux, ce qui démontre l'existence de l'automate déterministe $\mathcal{A}$ décrit dans la première étape.
□

Lemme 12.0.23. *Soit $k \geq 2$. Soient $t_{1,1}, t_{1,2}, t_{2,1}, t_{2,2}, s_1, s_2 \in \mathbb{S}^{\vec{n}}_{k+1}$ et $c_1, c_2 \in \mathbb{N}$. Considérons les suites doubles $\overline{f}, \overline{g}$ définies par :*

$$\overline{f}(m,n) = f(0,m,n), \quad \overline{g}(m,n) = g(0,m,n)$$

où les suites triples f, g vérifient, pour tous $0 \leq \ell \leq n$, $0 \leq m \leq n$

$$\begin{aligned}
f(n,n,n) &= c_1 \\
g(n,n,n) &= c_2 \\
f(\ell,m,n) &= f(\ell+1,m,n) \text{ si } \ell = m, \ell < n \\
g(\ell,m,n) &= g(\ell+1,m,n) \text{ si } \ell = m, \ell < n \\
f(n,m,n) &= c(n) \text{ si } 0 \leq m < n \\
g(n,m,n) &= d(n) \text{ si } 0 \leq m < n \\
f(\ell,m,n) &= b_{1,1}(\ell) f(\ell+1,m,n) + b_{1,2}(\ell) g(\ell+1,m,n) \text{ if } \ell \neq m, \ell < n \\
g(\ell,m,n) &= b_{2,1}(\ell) f(\ell+1,m,n) + b_{2,2}(\ell) g(\ell+1,m,n) \text{ si } \ell \neq m, \ell < n
\end{aligned}$$

Alors, $\overline{f}, \overline{g} \in \mathbb{S}^{(2),\vec{N}}_{k+1}$

Esquisse de preuve : Représentons chaque triplet (ℓ, m, n) où $0 \leq \ell \leq n, 0 \leq m \leq n$ par la 2-pile $\gamma_{l,m,n}$ suivante :

$$\begin{aligned}
\gamma_{\ell,m,n} &= a_2[{a_1}^{\ell}] \cdots a_2[{a_1}^{m-1}] \hat{\mathrm{A}}_2[{a_1}^{m}] a_2[{a_1}^{m+1}] \cdots a_2[{a_1}^{n-1}] \mathrm{A}_2[{a_1}^{n}] \text{ si } \ell \leq m < n, \\
\gamma_{\ell,m,n} &= a_2[{a_1}^{\ell}] \cdots a_2[{a_1}^{n-1}] \mathrm{A}_2[{a_1}^{n}] \text{ if } l \leq m < n, \\
\gamma_{\ell,m,n} &= a_2[{a_1}^{\ell}] \cdots a_2[{a_1}^{n-1}] \mathrm{B}_2[{a_1}^{n}] \text{ if } \ell \leq m = n.
\end{aligned}$$

où $\hat{\mathrm{A}}_2$ et B_2 sont des nouveaux symboles de niveau 2. La récurrence définissant f, g suit essentiellement le même schéma que celle du lemme 12.0.22 : c'est un schéma linéaire, où les deux variables sont ℓ, n tandis que m peut être vu comme un paramètre. Ces relations récurrentes peuvent être traduites en règles et finalement en un $(k+1)$-ACD. □

Lemme 12.0.24. *Si $f(m,n) \in \mathbb{S}^{(2),\vec{N}}_{k}$, $k \geq 0$, alors pour tous $m_0 \in \mathbb{N}$ fixé, $f(m_0,n)$, $f(n,n) \in \mathbb{S}^{\vec{N}}_{k}$.*

Preuve : Pour $f(m_0, n)$, il suffit de construire un automate permettant la dérivation

$$(q_0, a_{k+1}[T_{k,2}[{a_1}^n], q_0) \rightarrow^* (q_0, a_{k+1}T_{k,3}[\gamma_{0,n}], q_0) \rightarrow^{m_0} (q_0, b_{k+1}T_{k,3}[\gamma_{m_0,n}], q_0)$$

Puisque $f(m,n) \in \mathbb{S}_k^{(2),\vec{N}}$, il est possible de trouver un automate déterministe faisant aboutir ce calcul en $\alpha^{f(m_0,n)}$. Pour $f(n,n)$, c'est évident.

□

Lemme 12.0.25. *Soit $f(n) \in {\mathbb{S}_{k+1}}^{\vec{N}}$, $k \geq 2$ et F la suite double définie pour tous $0 \leq m \leq n$, par $F(m,n) = f(n)$. Alors F appartient à ${\mathbb{S}_{k+1}}^{(2),\vec{N}}$.*

Preuve : Il suffit de construire un automate permettant pour tous $m, n \geq 0$, la dérivation

$$(q_0, a_{k+1}[T_{k,3}[\gamma_{m,n}]], q_0) \rightarrow^* (q_0, a_{k+1}[T_{k,3}[b_2[{a_1}^n]]], q_0)$$

(on l'obtient par dépilements successifs du niveau 2) Par hypothèse $f \in \mathbb{S}_k^{\vec{N}}$, il est possible de trouver des transitions déterministes faisant aboutir ce calcul en $\alpha^{f(n)} = \alpha^{F(m,n)}$.

□

Lemme 12.0.26 (Produit mixte). *Soient $f \in {\mathbb{S}_3}^{\vec{N}}$, $g \in {\mathbb{S}_3}^{(2),\vec{N}}$, $k \geq 2$, alors $(f(m).g(m,n))_{0 \leq m \leq n} \in \mathbb{S}_3^{(2),\vec{N}}$*

Preuve : La preuve est analogue à celle de la proposition 11.5.5. Nous supposons que nous disposons d'un automate $\mathcal{A}$ réalisant les dérivations suivantes :
Règle d'initialisation, (I0) :

$$(q_0 a_3[\gamma_{m,n}]q_0) \rightarrow^*_{\mathcal{A}} (q_0 c_3[c_2[{a_1}^m]\gamma_{m,n}]q_0)$$

Calcul de f, (C1) :

$$(q_0, c_3[c_2[{a_1}^n]\Omega_2], q_0) \rightarrow^*_{\mathcal{A}} (q_0, b_3[\Omega_2], q_0)^{f(n)}$$

Calcul de g, (C2) :

$$(q_0, b_3[\gamma_{m,n}], q_0) \rightarrow^*_{\mathcal{A}} (q_0, \mathrm{A}_3[\varepsilon], q_0)^{g(m,n)}$$

Règle de terminaison, (T3) :

$$(q_0 \mathrm{A}_3[\varepsilon]q_0) \rightarrow_{\mathcal{A}} \alpha$$

Nous obtenons alors la dérivation suivante :

$$\begin{array}{rcll} (q_0 a_3[\gamma_{m,n}]q_0) & \rightarrow^*_{\mathcal{A}} & (q_0 c_3[c_2[{a_1}^n]\gamma_{m,n}]q_0) & (\text{par (I0)}) \\ & \rightarrow^*_{\mathcal{A}} & (q_0 b_3[\gamma_{m,n}]q_0)^{f(m)} & (\text{par (C1)}) \\ & \rightarrow^*_{\mathcal{A}} & (q_0 \mathrm{A}_3[\varepsilon]q_0)^{f(m)\cdot g(m,n)} & (\text{par (C2)}) \\ & \rightarrow^*_{\mathcal{A}} & \alpha^{f(m)\cdot g(m,n)} & (\text{par (T3)}) \end{array}$$

En procédant comme dans la preuve de la proposition 11.5.5, nous obtenons un automate déterministe réalisant ces dérivations.

□

Lemme 12.0.27 (Produit ordinaire). *Soient $f(m,n)$, $g(m,n) \in {\mathbb{S}_{k+1}}^{(2),\vec{N}}$, $k \geq 3$, alors* $(f(m,n).g(m,n))_{0\leq m\leq n} \in \mathbb{S}_{k+1}^{(2),\vec{N}}$

Preuve : La preuve est analogue à celle de la proposition 11.5.5. Nous supposons que nous disposons d'un automate $\mathcal{A}$ réalisant les dérivations suivantes :
Règle d'initialisation, (I0) : $\forall \omega_2 \in$ 2-Pile,

$$(q_0 a_{k+1}[T_{k,3}[\omega_2]]q_0) \rightarrow_{\mathcal{A}}^{*} (q_0 b_{k+1}[T_{k,3}[\omega_2]T_{k,3}[\omega_2]]q_0)$$

Calcul de f, (C1) :

$$(q_0, b_{k+1}[T_{k,3}[\gamma_{m,n}]\Omega_k], q_0) \rightarrow_{\mathcal{A}}^{*} (q_0, \mathrm{B}_{k+1}[\Omega_k], q_0)^{f(m,n)}$$

Calcul de g, (C2) :

$$(q_0, \mathrm{B}_{k+1}[T_{k,3}[\gamma_{m,n}]\Omega_k], q_0) \rightarrow_{\mathcal{A}}^{*} (q_0, \mathrm{A}_{k+1}[\Omega_k], q_0)^{g(m,n)}$$

Règle de terminaison, (T3) :

$$(q_0 \mathrm{A}_{k+1}[\varepsilon]q_0) \rightarrow_{\mathcal{A}} \alpha$$

Nous obtenons alors la dérivation suivante :

$$\begin{array}{rcll}
(q_0 a_{k+1}[T_{k,3}[\gamma_{m,n}]]q_0) \rightarrow_{\mathcal{A}}^{*} & (q_0 b_{k+1}[T_{k,3}[\gamma_{m,n}]T_{k,3}[\gamma_{m,n}]]q_0) & \text{(par (I0))} \\
\rightarrow_{\mathcal{A}}^{*} & (q_0 \mathrm{B}_{k+1}[T_{k,3}[\gamma_{m,n}]]q_0)^{f(m,n)} & \text{(par (C1))} \\
\rightarrow_{\mathcal{A}}^{*} & (q_0 \mathrm{A}_{k+1}[\varepsilon]q_0)^{f(m,n)\cdot g(m,n)} & \text{(par (C2))} \\
\rightarrow_{\mathcal{A}}^{*} & \alpha^{f(m,n)\cdot g(m,n)} & \text{(par (T3))}
\end{array}$$

En procédant comme dans la preuve de la proposition 11.5.5, nous obtenons un automate déterministe réalisant ces dérivations.
□

Lemme 12.0.28 (Pseudo-convolution). *Soient $f \in {\mathbb{S}_{k+1}}^{(2),\vec{N}}$ et $g \in \mathbb{S}_k^{(2),\vec{N}}$, $k \geq 3$. Alors la suite h définie pour tout $n \geq 0$ par $h(n) = \sum_{0\leq m\leq n} f(m,n)\cdot g(n-m,n)$, appartient à ${\mathbb{S}_{k+1}}^{\vec{N}}$.*

Preuve : La preuve est analogue à celle de la proposition 11.5.10, nous donnons juste les dérivations clés permettant de calculer h : Nous notons encore

$$T_{k,3}[\Omega_2] = a_k[\cdots[a_3[\Omega_2]]\cdots]$$

Supposons que $\mathcal{A}$ permette les dérivations de base suivantes :
Calcul de f, (D1) : pour tout $n, m \geq 0$

$$(q_0 b_{k+1}[T_{k,3}[\gamma_{m,n}]\Omega_k]q_0) \rightarrow_{\mathcal{A}}^{*} (q_0 \mathrm{B}_{k+1}[\Omega_k]q_0)^{f(m)}$$

Calcul de g, (D2) : pour tous $n \geq 0, m \in [0,n]$

$$(q_0 \mathrm{B}_{k+1}[T_{k,3}[\gamma_{n,m}]]q_0) \rightarrow_{\mathcal{A}}^{*} (q_0 \mathrm{A}_{k+1}[\varepsilon]q_0)^{g(m)}$$

Génération des couples, (G3) : pour tous $1 \leq m \leq n$

$$(q_0 a_{k+1}[T_{k,3}[\gamma_{m,n}]]q_0) \rightarrow^*_{\mathcal{A}} (q_0 a_{k+1}[T_{k,3}[\gamma_{m-1,n}]]q_0)(q_0 b_{k+1}[T_{k,3}[\gamma_{m,n}]T_{k,3}[\gamma_{m,n}]]q_0)$$

G30 : pour tout $n \geq 0$

$$(q_0 a_{k+1}[T_{k,3}[\gamma_{0,n}]]q_0) \rightarrow^*_{\mathcal{A}} (q_0 b_{k+1}[T_{k,3}[\gamma_{0,n}]T_{k,3}[\gamma_{0,n}]]q_0)$$

Règle de terminaison, (T4) :

$$(q_0 \mathrm{A}_{k+1}[\varepsilon]q_0) \rightarrow^*_{\mathcal{A}} \alpha.$$

La preuve que ces règles permettent de calculer h et la construction d'un automate déterministe réalisant ces règles sont identiques à celles de la preuve de la proposition 11.5.10.

□

Chapitre 13

Application à l'arithmétique faible

Dans [ER66], Elgot and Rabin exposent une méthode pour construire des prédicat unaires P sur l'ensemble des entiers pour lesquels la SOM-théorie de $\langle \mathbb{N}, +1, P\rangle$ est décidable ($+1$ désigne le relation binaire successeur). Différents résultats dans cette direction ont été établis depuis ([Sie70],[Sem84],[Mae99],[CT02]). Ce type de problèmes s'inscrit dans la perspective plus générale de l'étude des théories arithmétiques "faibles", qui possèdent d'intéressantes propriétés de décidabilité ([Bès01]).

Nous utilisons ici les résultats de décidabilité sur les automates à k-piles pour démontrer la décidabilité de la théorie monadique de structures $\langle \mathbb{N}, +1, P\rangle$, pour une vaste classe de prédicats P (théorèmes 13.1.3, 13.1.4 et 13.2.14) contenant par exemple $(n\lfloor\sqrt{n}\rfloor)_{n\in\mathbb{N}}$ ou $(n^2\lfloor\log n\rfloor)_{n\in\mathbb{N}}$. Ces résultats se généralisent également dans le cas de structures à plusieurs prédicats emboîtés (théorème 13.1.6), comme par exemple

$$\langle \mathbb{N}, +1, \{n^{k_m}\}_{n\geq 0}, \{n^{k_m k_{m-1}}\}_{n\geq 0}, \ldots, \{n^{k_1\cdots k_m}\}_{n\geq 0}\rangle,$$

pour $k_1, \ldots, k_m \geq 0$.

13.1 Extensions de la structure $\langle \mathbb{N}, +1\rangle$

Un graphe étiqueté sur un alphabet $\{\alpha, \beta_1, \ldots, \beta_m, e\}$ ne contenant qu'un unique chemin infini et sans boucle peut-être vu comme comme l'axe des entiers naturels marqué de certaines valeurs.

Définition 13.1.1 ($\mathbb{N}$-graphes). *Nous appelons $\mathbb{N}$-graphe, tout graphe $\mathcal{G} = (V, E)$, étiqueté sur un alphabet $\{\alpha, \beta_1, \ldots, \beta_m, e\}$, vérifiant :*
1- $\mathcal{G}$ est composé d'exactement un chemin, débutant d'un sommet v_0, étiqueté par un mot infini $u_\mathcal{G} \in \{\alpha, \beta_1, \ldots, \beta_m, e\}^\omega$
2- Le mot $u_\mathcal{G}$ contient une infinité d'occurrences des lettres α et $\beta_1, \ldots, \beta_m$.

Notons $v \xrightarrow{u}_\mathcal{G} v'$ pour signifier qu'il existe un chemin étiqueté par u, partant du sommet v et aboutissant en v', dans le graphe $\mathcal{G}$.

Un tel graphe définit un prédicat unaire sur les entiers : étant donné un $\mathbb{N}$-graphe $\mathcal{G}$, nous définissons l'injection $\varphi_\mathcal{G} : \mathbb{N} \to V$ et m prédicats $P_{1\mathcal{G}}, \ldots, P_{m\mathcal{G}} \subseteq \mathbb{N}$ de la façon suivante :

$$\text{il n'existe pas d'arc aboutissant dans } \varphi(0) \tag{43}$$

$$\forall n \in \mathbb{N}, \text{ il existe } u \in \{\beta_1, \ldots, \beta_m, e\}^* \cdot \alpha \text{ tel que } \varphi(n) \xrightarrow{u}_\mathcal{G} \varphi(n+1) \tag{44}$$

$$\forall n \in \mathbb{N},\ [n \in P_i \Leftrightarrow \exists v \in V, u \in \{e, \beta_1, \ldots, \beta_m\}^* \beta_i, \varphi(n) \overset{u}{\rightarrow\!\!\!\triangleright}_{\mathcal{G}} v]. \tag{45}$$

(Nous donnons un exemple figure 24 dans le cas où $m = 1$; les entiers vérifiant P_1 sont entourés d'un cercle). L'application φ est bien définie puisque le mot $u_{\mathcal{G}}$ contient une infinité

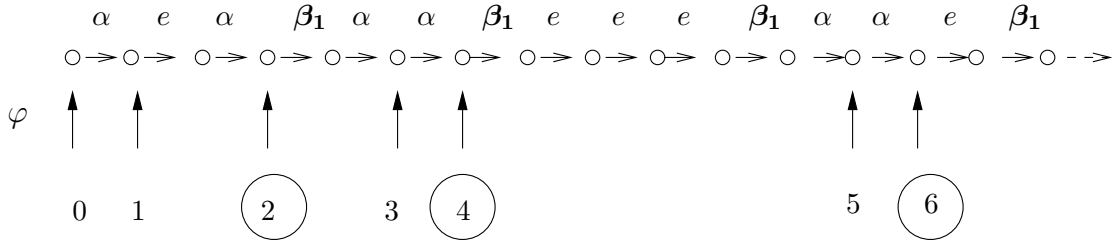


FIG. 24 – Un $\mathbb{N}$-graphe.

d'occurrences de α. Elle est injective puisque $\mathcal{G}$ n'est composé que d'un chemin.
Considérons la structure $\langle V, (R_a)_{a \in \{\alpha, \beta_1, \ldots, \beta_m, e\}} \rangle$ définie par

$$R_a = \{(v, v') \in V \times V, v \overset{a}{\rightarrow\!\!\!\triangleright}_{\mathcal{G}} v'\}.$$

Lemme 13.1.2. *Soit $\mathcal{G}$ un $\mathbb{N}$-graphe et soit φ , $P_1, \ldots, P_m$ l'application et les prédicats définis par (43)(44) (resp. (45)). Alors, l'application φ est une* SOM*-interprétation de la structure $\langle \mathbb{N}, S, P_1, \ldots, P_m \rangle$ dans la structure $\langle V, (R_a)_{a \in \{\alpha, \beta_1, \ldots, \beta_m, e\}} \rangle$.*

(Nous rappelons que les SOM-interprétations ont été introduites par la définition 3.1.3).
Preuve : Pour tout $v \in V, n, m \in \mathbb{N}$, les équivalences suivantes sont vraies :

$$v \in \mathrm{Im}(\varphi) \quad \Leftrightarrow \quad v \text{ n'a pas de prédécesseur ou } \exists v' \in V, v' \overset{\alpha}{\rightarrow\!\!\!\triangleright}_{\mathcal{G}} v \tag{46}$$

$$+_1(n, m) \quad \Leftrightarrow \quad \exists u \in \{\beta_1, \ldots, \beta_m, e\}^* \cdot \alpha,\ \varphi(n) \overset{u}{\rightarrow\!\!\!\triangleright}_{\mathcal{G}} \varphi(m) \tag{47}$$

$$P_i(n) \quad \Leftrightarrow \quad \exists v \in V, u \in \{\beta_1, \ldots, \beta_m, e\}^* \cdot \beta_i,\ \varphi(n) \overset{u}{\rightarrow\!\!\!\triangleright}_{\mathcal{G}} v \tag{48}$$

D'après la définition de φ et P, il est évident que le membre droit de l'équivalence (46) (resp. (47),(48)) peut être exprimé sous la forme $\Phi_1(v)$ (resp. $\Phi_2(\varphi(n), \varphi(m))$, $\Phi_3(\varphi(n))$) pour des SOM-formules Φ_1, Φ_2, Φ_3. Donc φ est une SOM-interprétation.
□

Théorème 13.1.3. *Pour toute suite $s \in \mathbb{S}_k$, la structure $\langle \mathbb{N}, +1, \Sigma s(\mathbb{N}) \rangle$ admet une* SOM*-théorie décidable.*

Preuve :
Cas 1 : s est ultimement nulle (i.e., il existe n_0 tel que $\forall n \geq n_0$, $s(n) = 0$).
Dans ce cas, Σs est ultimement constante et $\Sigma s(\mathbb{N})$ est exprimable dans la SOM-théorie de $\langle \mathbb{N}, +1 \rangle$. Ainsi, le théorème est vrai, par le théorème de Büchi ([Büc62]).
Cas 2 : s n'est pas ultimement nulle.
Considérons l'automate $\mathcal{A} \in k$-ACD construit dans la preuve du lemme 11.2.5. Soit $\mathcal{B} \in k$-ACD sur l'alphabet d'entrée $\{\alpha, \beta, e\}$ obtenu de $\mathcal{A}$ en remplaçant chaque ε-transition $(q, \varepsilon, w, \mathrm{instr}, q')$ par la transition $(q, e, w, \mathrm{instr}, q')$, et en recopiant les transitions lisant α ou β. Soit Graphe($\mathcal{B}$)

la structure associée au graphe de calcul de $\mathcal{B}$ (voir définition 2.5.2). D'après le lemme 11.2.5, $\mathsf{Graphe}(\mathcal{B})$ contient un chemin infini, débutant en $(q_0, \perp_k)$ et étiqueté par le mot infini :

$$u = u_0\beta u_1\beta \cdots \beta u_n\beta \cdots \text{ où } |u_n|_\alpha = s(n), \;\; u_n \in \{\alpha, e\}^* \tag{49}$$

De plus, $\mathsf{Graphe}(\mathcal{B})$ ne contient aucun arc sortant de ce chemin, et remplit donc la condition (1) de la définition 13.1.1. Puisque s est supposé non ultimement nulle, la forme spéciale du mot u donné en (49) assure que $\mathsf{Graphe}(\mathcal{B})$ vérifie la condition (2) de la définition 13.1.1. C'est donc un $\mathbb{N}$-graphe.
La forme du mot u dans (49) montre que le prédicat P associé à $\mathsf{Graphe}(\mathcal{B})$ est exactement $\Sigma s(\mathbb{N})$. D'après le lemme 13.1.2, $\langle \mathbb{N}, +1, \Sigma s(\mathbb{N}) \rangle$ est SOM-interprétable dans $\mathsf{Graphe}(\mathcal{B})$ qui admet une SOM-théorie décidable d'après le théorème 3.4.1, la structure $\langle \mathbb{N}, +1, \Sigma s(\mathbb{N}) \rangle$ admet une SOM-théorie décidable.
□

Théorème 13.1.4. *Si $s \in \mathbb{S}_k^{\vec{N}}$, avec $\vec{N} = (N_1, \ldots, N_m)$ tel que $\langle \mathbb{N}, +1, N_1, \ldots, N_m \rangle$ a une* SOM*-théorie décidable, alors $\langle \mathbb{N}, +1, \Sigma s(\mathbb{N}) \rangle$ a une* SOM*-théorie décidable.*

Preuve : En procédant comme dans la preuve du théorème 13.1.3, nous obtenons un automate $\mathcal{B} \in k\text{-}\mathsf{ACD}^{\vec{N}}$ tel que la structure $\langle \mathbb{N}, +1, \Sigma s(\mathbb{N}) \rangle$ est interprétable dans $\mathsf{Graphe}(\mathcal{B})$. D'après le théorème 6.2.2, l'hypothèse de décidabilité sur $\langle \mathbb{N}, +1, \vec{N} \rangle$, et le théorème 3.4.1, le graphe $\mathsf{Graphe}(\mathcal{B})$ admet une SOM-théorie décidable. Par SOM-interprétation, la structure S admet donc également une SOM-théorie décidable.
□

Corollaire 13.1.5. *Les structures $\langle \mathbb{N}, +1, \Sigma(n\lfloor\sqrt{n}\rfloor)_{n\in\mathbb{N}} \rangle$, et $\langle \mathbb{N}, +1, \Sigma(n\lfloor\log n\rfloor)_{n\in\mathbb{N}} \rangle$ admettent une* SOM*-théorie décidable.*

Preuve : Décrivons la preuve pour $n\lfloor\sqrt{n}\rfloor$. Considérons la suite s définie pour tout $n \geq 0$

$$\begin{cases} \lfloor\sqrt{n}\rfloor) & \text{si } n \notin \{m^2\}_{m\geq 0} \\ \lfloor\sqrt{n}\rfloor + n & \text{si } n \notin \{m^2\}_{m\geq 0}. \end{cases}$$

Alors $\Sigma s = (n\lfloor\sqrt{n}\rfloor)_{n\geq 0}$. En effet, pour tout $n \geq 0$:

$$(n+1)\lfloor\sqrt{n+1}\rfloor - n\lfloor\sqrt{n}\rfloor = \begin{cases} \lfloor\sqrt{n+1}\rfloor & \text{si } n+1 \notin \{m^2\}_{m\geq 0} \\ (n+1)(\lfloor\sqrt{n}\rfloor + 1) - n\lfloor\sqrt{n}\rfloor & \text{si } n+1 \in \{m^2\}_{m\geq 0} \end{cases}$$

Or si $n+1 \in \{m^2\}_{m\geq 0}$, alors, $(n+1)(\lfloor\sqrt{n}\rfloor + 1) - n\lfloor\sqrt{n}\rfloor = \lfloor\sqrt{n}\rfloor + n + 1 = \lfloor\sqrt{n+1}\rfloor + n$.

La suite s appartient à $\mathbb{S}_2^{\{m^2\}_{m\geq 0}}$. En effet en utilisant le lemme 11.4.7, nous il est possible de construire deux automate $\mathcal{A}_1$ et $\mathcal{A}_2 \in 2\text{-}\mathsf{ACD}^{\{m^2\}_{m\geq 0}}$ tels que $\mathcal{A}_1$ calcule $\lfloor\sqrt{n}\rfloor$ à partir de $(q_0, a_2[{a_1}^n])$ et $\mathcal{A}_2$ calcule $\lfloor\sqrt{n}+n\rfloor$ à partir de $(q_0, b_2[{a_1}^n])$. En supposant les alphabets de niveau 2 de ces deux automates sont disjoints, il est possible de construire $\mathcal{B} \in 2\text{-}\mathsf{ACD}^{\{m^2\}_{m\geq 0}}$ calculant $s(n)$ à partir de $(q_0, c_2[{a_1}^n])$ (c_2 est un nouveau symbole). Il suffit pour cela de choisir comme système de transition l'union de $\Delta_1 \cup \Delta_2$ avec les transitions

- $\Delta(q_0, \varepsilon, c_2a_1, 0) = (q_0, \varepsilon, c_2, 0) = (\text{change}_{a_2}, q_0)$
- $\Delta(q_0, \varepsilon, c_2a_1, 1) = (q_0, \varepsilon, c_2, 1) = (\text{change}_{b_2}, q_0)$.

La suite $(n\lfloor\sqrt{n}\rfloor)_{n\geq 0}$ appartient donc à $\Sigma\mathbb{S}_2^{\{m^2\}_{m\geq 0}}$ et la structure $\langle\mathbb{N}, +1, (n^2)_{n\in\mathbb{N}}\rangle$ admet une SOM-théorie décidable (voir par exemple [ER66]). En appliquant le théorème 13.1.4, nous prouvons donc que $\langle\mathbb{N}, +1, (n\lfloor\sqrt{n}\rfloor)_{n\geq 0}\rangle$ a une SOM-théorie décidable.

Pour le cas du logarithme, nous procédons de la même façon en utilisant le fait que $\langle\mathbb{N}, +1, (2^n)_{n\in\mathbb{N}}\rangle$ admet une SOM-théorie décidable (voir [ER66]).

□

Théorème 13.1.6. *Si $s \in \mathbb{S}_k^{\vec{N}}$, avec $\vec{N} = (N_1, \ldots, N_m)$ tel que $\langle\mathbb{N}, +1, N_1, \ldots, N_m\rangle$ a une* SOM*-théorie décidable, alors $\langle\mathbb{N}, +1, \Sigma s(\mathbb{N}), \Sigma s(N_1), \ldots, \Sigma s(N_m)\rangle$ a une* SOM*-théorie décidable.*

Preuve : Par une construction similaire à celle donnée dans la preuve du lemme 11.2.5, il est possible de construire un k-$\mathsf{ACD}^{\vec{N}}$ reconnaissant le langage $L \in (\{\alpha\} \cup \{\beta_{\vec{o}} \mid \vec{o} \in \{0,1\}^m\})^*$ suivant

$$L = \{\alpha^{s(0)}x_0 \cdots \alpha^{s(n)}x_n \mid n \geq 0, \forall i \in [1, n],\ x_i = \beta_{\chi_{\vec{N}}(i)}\}$$

Il suffit juste de remplacer chaque transition $(p, \beta, w, \vec{o}, \text{instr}, q)$ lisant β, par $(p, \beta_{\vec{o}}, w, \vec{o}, \text{instr}, q)$

En procédant comme dans le preuve du théorème 13.1.3, nous obtenons un automate $\mathcal{B} \in k$-$\mathsf{ACD}^{\vec{N}}$, dont le graphe de calcul est un $\mathbb{N}$-graphe et le chemin infini composant ce graphe est étiqueté (en supprimant les lettres e) par le mot infini

$$\alpha^{s(0)}\beta_{\chi_{\vec{N}}(0)} \cdots \alpha^{s(n)}\beta_{\chi_{\vec{N}}(n)} \cdots$$

Posons $P_{\vec{o}} = \{n \mid \chi_{\vec{N}}(n) = \vec{o}\}$. D'après le lemme 13.1.2, la structure $S = \langle\mathbb{N}, +1, \Sigma s(\mathbb{N}), (\Sigma s(P_{\vec{o}}))_{\vec{o}\in\{0,1\}^m}\rangle$ est SOM-interprétable dans $\mathsf{Graphe}(\mathcal{B})$. D'après le théorème 6.2.2, l'hypothèse de décidabilité sur $\langle\mathbb{N}, +1, \vec{N}\rangle$, et le théorème 3.4.1, le graphe $\mathsf{Graphe}(\mathcal{B})$ admet une SOM-théorie décidable. Par SOM-interprétation, la structure S admet donc également une SOM-théorie décidable.

Enfin, la structure $\langle\mathbb{N}, +1, \Sigma s(\mathbb{N}), \Sigma s(N_1), \ldots, \Sigma s(N_m)\rangle$ est clairement SOM-interprétable dans S puisque pour tout $i \in [1, m]$,

$$\Sigma s(N_i) = \bigcup_{\vec{o}\mid\pi_i(\vec{o})=1} \Sigma s(P_{\vec{o}})$$

et admet donc une SOM-théorie décidable.

□

Corollaire 13.1.7. *Pour tous entiers $k_1, \ldots, k_m \geq 0$, la* SOM*-théorie de la structure*

$$\langle\mathbb{N}, +1, \{n^{k_m}\}_{n\geq 0}, \{n^{k_m k_{m-1}}\}_{n\geq 0}, \ldots, \{n^{k_1 \cdots k_m}\}_{n\geq 0}\rangle$$

est décidable.

Preuve : Posons $u_i(n) = \sum_{j=0}^{k_i} \binom{j}{k} n^j$. Nous avons clairement $\Sigma u_i(n) = n^{k_i}$ et d'après le théorème 11.5.19, $u_i \in \mathbb{S}_2$ puisque c'est une suite $\mathbb{N}$-rationnelle. Donc, pour tout $i \in [1, m]$, la suite $(n^{k_i})_{n\in\mathbb{N}}$ appartient à $\Sigma\mathbb{S}_2$.

Vérifions la validité du corollaire par induction sur $m \geq 1$.

Base : Si $m = 1$, alors puisque $(n^{k_1})_{n\in\mathbb{N}}$ appartient à $\Sigma\mathbb{S}_2$, le théorème 13.1.3 implique directement que la décidabilité de la théorie de $\langle\mathbb{N}, +1, \{n^{k_1}\}_{n\geq 0}\rangle$.

Pas d'induction : Supposons le corollaire vrai pour $m \geq 1$. La suite u_{m+1} appartient à $\mathbb{S}_2$. Posons $\vec{N} = (N_1, \ldots, N_m)$ avec pour tout $i \in [1, m]$

$$N_i = \{n^{k_m \cdots k_i} \mid n \geq 0\}$$

La suite u_{m+1} appartient à ${\mathbb{S}_2}^{\vec{N}}$ et le théorème 13.1.6 implique donc que

$$\text{la SOM-théorie de } \langle \mathbb{N}, +1, \Sigma u_{m+1}(\mathbb{N}), \Sigma s(N_1), \ldots, \Sigma s(N_m) \rangle \text{ est décidable}$$

De plus $\Sigma u_{m+1}(\mathbb{N}) = \{n^{k_{m+1}}\}_{n \geq 0}$ et pour tout $i \in [1, m]$,

$$\Sigma u_{m+1}(N_i) = \{ \sum_{j=0}^{j=n^{k_m \cdots k_i}} u_{m+1}(j) \mid n \geq 0\} = \{(n^{k_m \cdots k_i})^{k_{m+1}} \mid n \geq 0\},$$

ce qui prouve le corollaire.

□

13.2 Suites à différences k-calculables

La forme particulière du prédicat $\Sigma s(\mathbb{N})$ considéré dans le théorème 13.1.3 amène naturellement à l'étude de la classe de suites suivante.

Définition 13.2.1. *Soit $k \geq 2$ et $\vec{N}$ un vecteur de sous-ensembles de $\mathbb{N}$. Nous définissons la classe $\Sigma\mathbb{S}_k \subseteq \mathbb{N}^{\mathbb{N}}$ comme l'ensemble*

$${\Sigma\mathbb{S}_k}^{\vec{N}} = \{\Sigma s \mid s \in {\mathbb{S}_k}^{\vec{N}}\}$$

Nous étudions ici les propriétés de clôture de la classe ${\Sigma\mathbb{S}_k}^{\vec{N}}$. Nous débutons par prouver que cette classe appartient à une famille remarquable de suite généralisant la famille des suites *résiduellement ultimement périodiques* (RUP) étudiée dans [CT02].

Définition 13.2.2 (Suites résiduellement $\vec{N}$-définissables). *Une suite $(s_n)_{n \geq 0}$ est dite* $\vec{N}$-RD *si pour tout semi-groupe fini $(S, \star)$ et tout morphisme $\mu : \mathbb{N} \to S$, le langage $L_{\mu,s} = \{\mu(s_0) \cdots \mu(s_n) \mid n \geq 0\}$ est SOM-définissable dans la structure $\langle S^*, (\text{SUCC}_s)_{s \in S}, {|\vec{N}|_S}^{-1} \rangle$.*

En particulier, si u est $\vec{\emptyset}$-RD, alors $L_{\mu,u}$ est régulier, et puisque ce langage est préfixe, le mot infini $\mu(u_0) \cdots \mu(u_n) \cdots$ est périodique à partir d'une certaine longueur (il existe $\sigma, \gamma \in S^*$ tels que $\mu(u_0) \cdots \mu(u_n) \cdots = \gamma \cdot \sigma^\omega$). En d'autres termes, la suite s est RUP.

Proposition 13.2.3. *Si la structure $\langle \mathbb{N}, +1, \vec{N} \rangle$, vérifie la propriété* MD*, alors pour toute suite $s \in \mathbb{S}_k^{\vec{N}}$, $k \geq 1$, les suite s et Σs sont $\vec{N}$-RD.*

Preuve : Soit $(S, \star, e)$ un semi-groupe fini, $\mu : \mathbb{N} \to S$ un morphisme et $(u_n)_{n \geq 0}$ une suite dans ${\mathbb{S}_{k+1}}^{\vec{N}}$, pour calculée par l'automate $\mathcal{A} \in k\text{-}\mathsf{ACD}^{\vec{N}}$.

Nous modifions cet automate de façon à calculer dans les états les valeurs de $\mu(u_n)$ et à autoriser les piles à contenir des éléments de S au niveau 1. Il suffit principalement d'effectuer sur les transitions de $\mathcal{A}$ les modifications suivantes :

– chaque $(p, \alpha, w, q, \text{instr})$ est substituée par l'ensemble $((p, s), \alpha, w, \text{instr}, (q, s \star \mu(1)))$, $\forall s \in S$,
– et chaque transition $(p, \varepsilon, w, \text{instr}, q)$ est substituée par $((p, s), \varepsilon, w, \text{instr}, (q, s))$, $\forall s \in S$.

Il ne reste ensuite plus qu'à remplacer dans chaque transition la liste de symboles de tête wa_1 par ws pour tout $s \in S$, et de remplacer chaque instruction push_{a_1} par push_s (le symbole choisi s n'a pas d'importance, nous pouvons par exemple fixer par convention que nous empilons toujours le symbole $\mu(0)$).

Nous obtenons ainsi un nouvel automate déterministe $\mathcal{A}_1 \in k\text{-}\mathsf{NAP}^{(l_S{}^{-1}(\vec{N}))_1}$ vérifiant pour tout mot $s_1 \cdots s_n s_{n+1} \in S^*$, $n \geq 0$:

$$((q_0, \mu(0)), \alpha^{u(n)}, T_{k,2}[s_1 \cdots s_n]) \vdash_{\mathcal{A}}^* ((q_0, \mu(u(n))), \varepsilon, \varepsilon)$$

et

$$((q_0, \mu(\Sigma u(n))), \alpha^{u(n+1)}, T_{k,2}[s_1 \cdots s_{n+1}]) \vdash_{\mathcal{A}_1}^* ((q_0, \mu(\Sigma u(n+1))), \varepsilon, \varepsilon),$$

où comme usuellement $T_{k,2}[\Omega_1]$ désigne le terme $a_k[a_{k-1}[\cdots[a_2[\Omega_1]]\cdots]]$.

Nous définissons à partir de $\mathcal{A}_1$ deux nouveaux automates déterministes en reprenant le même type de construction que pour le lemme 11.2.5. Posons

$\mathcal{B}_1 = (Q_1 \cup \{q_F\} \times S, \{\alpha\}, (S, A_2, \ldots, A_k \cup \{b_k\}), \Delta_1 \cup \Delta, (l^{-1}(\vec{N})_1, (q_0, \mu(0)), \{q_F\})$, et

$\mathcal{B}_2 = (Q_1 \cup \{q_F\} \times S, \{\alpha\}, (S, A_2, \ldots, A_k \cup \{b_k\}), \Delta_2 \cup \Delta, (l^{-1}(S))_1, (q_0, \mu(0)), \{q_F\})$, où Q_1 est l'ensemble des états de $\mathcal{A}_1$, Δ est la relation de transition de $\mathcal{A}_1$ et Δ_1, Δ_2 sont obtenus de la façon suivante :

pour tout $\vec{o} \in \{0,1\}^{|\vec{N}|}$,
– $((q_0, \mu(0)), \varepsilon, \varepsilon, \vec{o}, \text{push}_{b_k}\, \text{push}_{a_{k-1}} \cdots \text{push}_{a_2}\, \text{push}_{a_k}, (q_0, \mu(0))) \in \Delta_1, \Delta_2$,
– pour tous $s, s' \in S$,
  – $\Delta_1((q_0, s), \varepsilon, b_k a_{k-1} \cdots a_2, \vec{o}) = \Delta_1((q_0, s), \varepsilon, b_k a_{k-1} \cdots a_2 s', \vec{o}) = (\text{push}_{1,s}, (q_F, \mu(0)))$,
  – $\Delta_2((q_0, s), \varepsilon, b_k a_{k-1} \cdots a_2, \vec{o}) = \Delta_2((q_0, s), \varepsilon, b_k a_{k-1} \cdots a_2 s', \vec{o}) = (\text{push}_{1,s}, (q_F, s))$,
– pour tout $s, s' \in S$,
  $((q_F, s), \varepsilon, b_k a_{k-1} \cdots a_2 s', \vec{o}, \text{push}_{a_k}, (q_0, s)) \in \Delta_1, \Delta_2$.

Ces deux automates sont déterministes, et une simple induction permet de vérifier que
$\text{ACC}(\mathcal{B}_1) = \{b_k[a_{k-1} \cdots [a_2[\mu(u_n) \cdots \mu(u_0)]] \cdots]),\ n \geq 1\}$, et
$\text{ACC}(\mathcal{B}_2) = \{b_k[a_{k-1} \cdots [a_2[\mu(\Sigma u_n) \cdots \mu(\Sigma u_0)]] \cdots]),\ n \geq 1\}$.

En particulier,

$$\varphi_1(\text{p}_1(\text{ACC}(\mathcal{B}_1))) = L_{\mu,u} \text{ et } \varphi_1(\text{p}_1(\text{ACC}(\mathcal{B}_2))) = L_{\mu,\Sigma u}.$$

Or, $\text{ACC}(\mathcal{B}_1)$ et $\text{ACC}(\mathcal{B}_2)$ appartiennent à $\mathsf{PREC}_k{}^{(l^{-1}(\vec{N}))_1}$, et donc en appliquant itérativement la proposition 6.2.5(2),

$$\text{p}_1(\text{ACC}(\mathcal{B}_1)) \text{ et } \text{p}_1(\text{ACC}(\mathcal{B}_2)) \in \mathsf{PREC}_1{}^{\vec{N}}.$$

C'est à dire, par définition de $\mathsf{PREC}_k{}^{\vec{N}}$ que $L_{\mu,u}$ et $L_{\mu,\Sigma u}$ appartiennent à $\mathsf{REC}_1^{\vec{N}}(S)$.

□

Remarque 13.2.4. *Il est prouvé dans [CT02] que pour toute suite s RUP, la théorie* SOM *de la droite des entiers naturels augmentée du prédicat $s(\mathbb{N})$ admet une théorie* SOM *décidable. Les classes $\Sigma\mathbb{S}_k$ sont donc incluses dans celle étudiée dans [CT02]. Par contre, les suites dans $\Sigma\mathbb{S}_k^{\vec{N}}$ pour lesquelles l'extension de la droite a une théorie* SOM *décidable (voir théorème 13.1.6, comme par exemple $(n\lfloor\log(n)\rfloor)_{n\in\mathbb{N}}$ et $(n\lfloor\log(n)\rfloor)_{n\in\mathbb{N}}$ sortent de la classe des suites RUP.*

Proposition 13.2.5. *Il existe une suite s dans $\Sigma\mathbb{S}_k^{\vec{N}}$ telle que $\langle \mathbb{N}, +1, s(\mathbb{N})\rangle$ admet une théorie SOM décidable et qui n'est pas RUP.*

Preuve : Considérons la suite $s = \lfloor \sqrt{n} \rfloor$. En procédant comme dans la preuve du corollaire 13.1.5, il est possible de prouver que la théorie SOM de la droite infini augmentée de $s(\mathbb{N})$ admet un théorie SOM décidable.

Soit μ_2 le morphisme $\mathbb{N} \to \{0,1\}$ défini par $n \mapsto (n)_{mod2}$. Alors la suite $\mu_2(s)$ est définie pour tout $n \geq 0$ par

$$\mu_2(s(n)) = \begin{cases} 0 & \text{si } \exists k \geq 0,\ n \in [(2k)^2,\ (2k+1)^2 - 1] \\ 1 & \text{si } \exists k \geq 0,\ n \in [(2k+1)^2,\ (2k)^2 - 1] \end{cases}$$

Il est donc clair qu'il n'existe aucun $p \geq 0$ tel que le mot infini $\mu_2(s(p))\mu_2(s(p+1))\cdots$ est périodique puisque les plages de 0 et 1 ne cessent de grandir.

□

Nous montrons que maintenant que les classes $\Sigma\mathbb{S}_k^{\vec{N}}$ sont stables par de nombreuses opérations. Les différents opérateurs sur les suites et séries sont définis dans §IV.

Fait 13.2.6. *Soit $k \geq 2$ et $u \in \mathbb{N}^{\mathbb{N}}$. La suite u appartient à $\Sigma\mathbb{S}_k^{\vec{N}}$ si et seulement si Δu appartient à $\mathbb{S}_k^{\vec{N}}$.*

Preuve : Les deux identités suivantes sont évidentes :

$$\Delta\Sigma u = \mathsf{E}u; \quad \Sigma\Delta u = \mathsf{E}u - \frac{u(0)}{1-X}.$$

1- si u appartient à $\Sigma\mathbb{S}_k$, $u = \Sigma v$ pour $v \in \mathbb{S}_k$. Alors $\Delta u = \Delta\Sigma v = \mathsf{E}v$, et d'après le théorème 11.5.19(0), $\mathsf{E}v \in \mathbb{S}_k$.

2- Supposons que $\Delta u \in \mathbb{S}_k$. Alors $\Sigma\Delta u + \frac{u(0)}{1-X} = \mathsf{E}u$. De façon équivalente, $\Sigma(\Delta u + u(0)) = \mathsf{E}u$. Soit $v_0 = u_0$ et $v_{n+1} = (\Delta u)_n$. D'après le théorème 11.5.19, point 0, v appartient à $\mathbb{S}_k$ et d'après la formule ci-dessus $u = \Sigma v$. Donc $u \in \Sigma\mathbb{S}_k$.

□

Lemme 13.2.7. *Soit $k \geq 1$ et $U \in \Sigma\mathbb{S}_{k+1}^{\vec{N}}$. Alors $\mathsf{E}U \in \Sigma\mathbb{S}_{k+1}^{\vec{N}}$.*

Preuve : Supposons que $U \in \Sigma\mathbb{S}_{k+1}^{\vec{N}}$. Notons que $\Delta\mathsf{E}U = \mathsf{E}\Delta U$. En utilisant le fait 13.2.6 et la stabilité de $\mathbb{S}_{k+1}^{\vec{N}}$ par décalage, nous obtenons que $\mathsf{E}U \in \Sigma\mathbb{S}_{k+1}^{\vec{N}}$.

□

Lemme 13.2.8. *Soit $k \geq 1$ et $U, V \in \Sigma\mathbb{S}_{k+1}^{\vec{N}}$. Alors $U + V \in \Sigma\mathbb{S}_{k+1}^{\vec{N}}$.*

Lemme 13.2.9. *Soit $k \geq 2$ et $U, V \in \Sigma\mathbb{S}_{k+1}^{\vec{N}}$. Alors $U \odot V \in \Sigma\mathbb{S}_{k+1}^{\vec{N}}$.*

Preuve : Soient $U, V \in \Sigma\mathbb{S}_{k+1}^{\vec{N}}$. L'identité suivante est bien connue :

$$\Delta(U \odot V) = \Delta(U \odot \mathsf{E}V) + U \odot (\Delta V).$$

En utilisant le théorème 11.5.19, les suites $U, V, \mathsf{E}V$ appartiennent toutes à $\mathbb{S}_{k+1}^{\vec{N}}$, et donc le membre droit de l'identité précédente doit appartenir à $\mathbb{S}_{k+1}^{\vec{N}}$. Par le fait 13.2.6, nous obtenons $U \odot V \in \Sigma\mathbb{S}_{k+1}^{\vec{N}}$.

□

Lemme 13.2.10. *Soit $k \geq 2$ et $U \in \Sigma\mathbb{S}^{\vec{N}}_{k+1}$, $V \in \Sigma\mathbb{S}_k$. Alors $U \times V \in \Sigma\mathbb{S}^{\vec{N}}_{k+1}$.*

Preuve : Soit $U = \Sigma u$ et $V = \Sigma v$ pour $u \in \mathbb{S}^{\vec{N}}_{k+1}$ et $v \in \mathbb{S}_k$. Transformons l'expression $\Delta(U \times V)$ en une expression qui n'utilise plus l'opérateur Δ.

$$\begin{aligned}
\Delta(\Sigma u \times \Sigma v) &= \frac{(\Sigma u \times \Sigma v)(1-X) - \Sigma u(0) \cdot \Sigma v(0)}{X} \\
&= \frac{\frac{u \times v}{1-X} - \frac{u(0) \cdot v(0)}{1-0}}{X} \\
&= \mathsf{E}(\frac{u \times v}{1-X}) \\
&= \mathsf{E}\Sigma(u \times v).
\end{aligned}$$

En utilisant le théorème 11.5.19(3), l'expression finale obtenue appartient à $\mathbb{S}^{\vec{N}}_{k+1}$, ce qui prouve que $\Sigma u \times \Sigma v$ appartient à $\Sigma\mathbb{S}^{\vec{N}}_{k+1}$.

□

Lemme 13.2.11. *Soit $V \in \Sigma\mathbb{S}_k$, $k \geq 2$, tel que $V(0) \geq 1$. Soit U la suite définie par*

$$U(0) = 1 \text{ et pour tout } n \geq 0, U(n+1) = \sum_{k=0}^{n} U(k) \cdot V(n-k).$$

Alors $U \in \Sigma\mathbb{S}_{k+1}$.

Preuve : Soit $v \in \mathbb{S}_{k+1}$ tel que $V = \Sigma v$. Comme remarqué en 11.5.14

$$U = \frac{1}{1 - \frac{Xv}{1-X}}.$$

Calculons la série ΔU.

$$\Delta U = \Delta(\frac{1}{1 - \frac{Xv}{1-X}}) = \Delta(\frac{1-X}{1-X-Xv}) = \frac{1}{X}[\frac{(1-X)^2}{1-X-Xv} - 1]$$

donc

$$\Delta U = \frac{1}{X}[\frac{(-X + X^2 + Xv)}{1-X-Xv}] \tag{50}$$

Calculons la série $U \times v - 1$:

$$U \times v - 1 = \frac{(1-X)v}{1-X-Xv} - 1 = \frac{1}{X}[\frac{(-X + X^2 + Xv)}{1-X-Xv}] \tag{51}$$

À partir des équations (50),(51) nous obtenons l'identité :

$$\Delta U = U \times v - 1. \tag{52}$$

Par les propriétés de stabilité établies dans le théorème 11.5.19, U appartient à $\mathbb{S}_{k+1}$ et $U \times v$ appartient également à $\mathbb{S}_{k+1}$. L'hypothèse $U(0) = 1$ et $V(0) \geq 1$ assure que $U \times v - 1$ appartient à $\mathbb{S}_{k+1}$. La formule (52) montre que $U \in \Sigma\mathbb{S}_{k+1}$.

□

Lemme 13.2.12. *Soit $k_1 \geq 1, k_2 \geq 1, U \in \Sigma\mathbb{S}_{k_1+1}$ et $V \in \Sigma\mathbb{S}^{\vec{N}}_{k_2+1}$. Alors $U \circ V \in \Sigma\mathbb{S}^{\vec{N}}_{k_1+k_2+1}$.*

Esquisse de preuve : Soit $U = \Sigma u, V = \Sigma v$ pour $u \in \mathbb{S}_{k_1+1}, v \in \mathbb{S}^{\vec{N}}_{k_2+1}$. Alors $U \circ V = \sum_{m=0}^{m=V(n)} u(m)$. Donc

$$(\Delta(U \circ V))(n) = \sum_{m=V(n)+1}^{m=V(n+1)} u_m$$

Prouvons donc que $\sum_{m=V(n)+1}^{m=V(n+1)} u_m$ appartient à $\mathbb{S}^{\vec{N}}_{k_1+k_2+1}$.

Posons $k = k_1 + k_2 + 1$. Un k-ACD calculant $U \circ V$ peut être construit en suivant les lignes suivantes.

Notons que, d'après le théorème 11.5.19, point (6), V appartient également à $\mathbb{S}_{k_2+1}$. Par le lemme 11.5.1, il existe un (k_1+1)-ACD $\mathcal{A}$ sur les alphabets de piles $A_{k_1+1} \supseteq \{a_{k_1+1}, \mathrm{A}_{k_1+1}\}$ et $A_i \supseteq \{a_i\}$ pour $i \in [1, k_1]$, il existe un (k_2+1)-ACD $\mathcal{B}$ sur les alphabets de piles $B_{k_2+1} \supseteq \{b_{k_2+1}, \mathrm{B}_{k_2+1}\}$ et $B_i \supseteq \{b_i\}$ et il existe un (k_2+1)-ACD $\mathcal{C}$ sur les alphabets de piles $C_{k_2+1} \supseteq \{c_{k_2+1}, \mathrm{C}_{k_2+1}\}$ et $C_i \supseteq \{c_i\}$ pour $i \in [1, k]$, $b_1 = c_1$ avec les ensembles d'états $Q_{\mathcal{A}} \ni q_0, Q_{\mathcal{B}}, Q_{\mathcal{C}}$, choisis tels que :

$$\begin{aligned}
Q_{\mathcal{B}} \cap Q_{\mathcal{C}} &= \{r_0\} \\
(q_0 a_k[a_{k-1}[\cdots[{a_{k_2+1}}^n]]\cdots]q_0) &\to^*_{\mathcal{A}} (q_0 \mathrm{A}_k[\varepsilon]q_0)^{u(n)} && (53)\\
(r_0 b_{k_2+1}[b_{k_2}[\cdots[b_2[{b_1}^n]]\cdots]r_0) &\to^*_{\mathcal{B}} (r_0 \mathrm{B}_{k_2+1}[\varepsilon]r_0)^{v(n)} && (54)\\
(r_0 c_{k_2+1}[c_{k_2}[\cdots[c_2[{c_1}^n]]\cdots]r_0) &\to^*_{\mathcal{C}} (r_0 \mathrm{C}_{k_2+1}[\varepsilon]r_0)^{V(n)}. && (55)
\end{aligned}$$

La dérivation (54) prouve l'existence d'une suite $H_1, \ldots, H_{v(n)}$ de (k_2+1)-Termes vérifiant :

$$H_{v(n+1)} = b_{k_2+1}[b_{k_2}[\cdots[b_2[{b_1}^{n+1}]]\cdots]\Omega_{k_2}], \quad H_1 = \mathrm{B}_{k_2+1}[\Omega_{k_2}],$$

$$(r_0 H_{i+1} r_0) \to_{\mathcal{B}} (r_0 \mathrm{B}_{k_2+1}[\Omega_{k_2}]r_0)(r_0 H_i[\Omega_{k_2}]r_0).$$

Posons $\gamma_n = d_{k_1+1}[\cdots[d_2[{d_1}^n]]\cdots]$. Par une construction analogue à celle donnée proposition 11.5.16, nous obtenons un (k)-ACD $\mathcal{D}$, sur les alphabets de piles $D_i \supseteq B_i \cup C_i$, pour tout $i \in [1, k_2+1]$, $D_{k_2+i} \supseteq A_i$, pour $i \in [1, k_1+1]$ et D_k contient également les nouveau symboles d_k et D_k, et l'ensemble d'états $Q_{\mathcal{D}} \supseteq Q_{\mathcal{A}} \times (Q_{\mathcal{B}} \cup Q_{\mathcal{C}})$, et rendant les règles suivantes valides :

Génération des arguments, (G1) : pour tout $n \geq 0$

$$((q_0, r_0) d_k[T_{k-1,k_1+1}[\gamma_n]](q_0, r_0)) \to^*_{\mathcal{D}} \prod_{i=1}^{v(n+1)} ((q_0, r_0) T_{k,k_1+1}[H_i \cdot T_{k_1+1,2}[{a_1}^n]](q_0, r_0))$$

Calcul de $u \circ v$, (C2) : pour tout $n \geq 0, v(n+1) \geq i \geq 1$

$$(q_0, r_0) T_{k,k_1+2}[H_i \cdot T_{k_1+1,2}[{a_1}^n]](q_0, r_0) \to^*_{\mathcal{D}} ((q_0, r_0) \mathrm{D}_k[\varepsilon](q_0, r_0))^{u(i+V(n))}.$$

En composant (G1) et (C2) nous obtenons finalement :

$$\begin{aligned}
(q_0, r_0) d_k[T_{k-1,k_1}[\gamma_n]](q_0, r_0) &\to^*_{\mathcal{D}} \prod_{i=1}^{v(n+1)} ((q_0, r_0) \mathrm{D}_k[\varepsilon](q_0, r_0))^{u(i+V(n))} \\
&= ((q_0, r_0) \mathrm{D}_k[\varepsilon](q_0, r_0))^{\Delta(U \circ V)(n)}
\end{aligned}$$

□

Lemme 13.2.13. *Soit $k \geq 2$. Soient $U_1, U_2, \ldots, U_p$ des suites d'entiers, $P_1, P_2, \ldots, P_p$ des polynômes dans $\Sigma\mathbb{S}^{\vec{N}}_{k+1}[X_1, X_2, \ldots, X_p]$, $c_1, c_2, \ldots, c_p \in \mathbb{N}$ tels que : pour tout $1 \leq i \leq p$*

$$U_i(n+1) = P_i(n, U_1(n), U_2(n), \ldots, U_p(n)) \text{ et } c_i = U_i(0) \leq U_i(1).$$

Alors $U_1 \in \Sigma\mathbb{S}^{\vec{N}}_{k+1}$.

Preuve : Soit U_i, P_i, c_i vérifiant l'hypothèse du lemme. Soit $a_0(n), a_1(n), \ldots, a_q(n)$ une suite énumérant les coefficients des polynômes $P_1, P_2, \ldots, P_p$.

Il existe des polynômes $Q_i \in \mathbb{N}[X_0, \ldots, X_{q+p}]$, tels que, pour tout $i \in [1, p]$:

$$P_i(n, U_1(n), U_2(n), \ldots, U_p(n)) = Q_i(a_0(n), \ldots, a_q(n), U_1(n), \ldots, U_p(n)).$$

La formule de Euler-Mac-Laurin appliquée aux polynômes Q_i exprime la différence

$$Q_i(X_0, \ldots, X_{q+p}) - Q_i(Y_0, \ldots, Y_{q+p})$$

sous la forme :

$$\sum_{\bar{k}} \frac{1}{\bar{k}!} \frac{\partial^{\bar{k}} Q_i}{(\partial X_0)^{k_0} \ldots (\partial X_{q+p})^{k_{q+p}}}(Y_0, \ldots, Y_{q+p}) \cdot (X_0 - Y_0)^{k_0} \cdots (X_{q+p} - Y_{q+p})^{k_{q+p}}, \tag{56}$$

où $\bar{k} = (k_1, k_2, \ldots, k_{q+p})$ varie sur tous les $(q+p)$-tuples avec une somme $k_1 + k_2 + \ldots + k_{q+p}$ inférieure ou égale au degré de Q_i. Pour tout monôme $M = X_0^{d_0} X_1^{d_1} \cdots X_{q+p}^{d_{q+p}}$ la dérivée partielle

$$\frac{1}{\bar{k}!} \frac{\partial^{\bar{k}} M}{(\partial X_0)^{k_0} \ldots (\partial X_{q+p})^{k_{q+p}}}(Y_0, \ldots, Y_{q+p}),$$

est égale à

$$\binom{d_0}{k_0} \binom{d_1}{k_1} \cdots \binom{d_{q+p}}{k_{q+p}} \cdot Y_0^{d_0 - k_0} Y_1^{d_1 - k_1} \cdots Y_{q+p}^{d_{q+p} - k_{q+p}}. \tag{57}$$

Toute dérivée partielle

$$R_{i,\bar{k}} = \frac{\partial^{\bar{k}} Q_i}{(\partial X_0)^{k_0} \ldots (\partial X_{q+p})^{k_{q+p}}}(Y_0, \ldots, Y_{q+p})$$

est une combinaison linéaire, avec coefficients dans $\mathbb{N}$, de monômes de la forme (57), donc, elle ne possède que des coefficients entier non négatifs :

$$R_{i,\bar{k}} \in \mathbb{N}[Y_0, \ldots, Y_{q+p}].$$

Appliquons la substitution suivante aux indéterminées $X_0, \ldots, X_{q+p}, Y_0, \ldots, Y_{q+p}$,

$$X_j \leftarrow a_j(n+1) \text{ pour } 0 \leq j \leq q; \;\; X_{q+\ell} \leftarrow U_\ell(n+1) \text{ pour } 0 \leq \ell \leq p,$$

$$Y_j \leftarrow a_j(n) \text{ pour } 0 \leq j \leq q; \;\; Y_{q+\ell} \leftarrow U_\ell(n) \text{ pour } 0 \leq \ell \leq p.$$

Nous obtenons : $(\Delta U_i)(n+1) =$

$$\sum_{\bar{k}} R_{i,\bar{k}}(a_0(n+1), \ldots, a_q(n+1), U_1(n), \ldots, U_p(n)) \cdot (\Delta \bar{a}(n))^{\bar{k}} \cdot (\Delta \bar{U}(n))^{\bar{k}} \tag{58}$$

où l'expression $(\Delta\bar{a})^{\bar{k}}(n)$ signifie :

$$(\Delta a_0)^{k_0}(n)\cdots(\Delta a_q)^{k_q}(n)$$

et l'expression $(\Delta\bar{U})^{\bar{k}}(n)$ signifie :

$$(\Delta U_1)^{k_{q+1}}(n)\cdots(\Delta U_p)^{k_{q+p}}(n).$$

Par les propriétés de clôtures établies dans le théorème 11.5.19, toute suite $R_{i,\bar{k}}(a_0(n+1),\ldots,a_q(n+1),U_1(n),\ldots,U_p(n))$ appartient à $\mathbb{S}^{\vec{N}}_{k+1}$. L'équation (58) est donc un système de polynômes à coefficients dans $\mathbb{S}_{k+1}$, avec conditions initiales $U_i(1)-U_i(0)\in\mathbb{N}$ et dont le vecteur solution est :

$$((\Delta U_1)(n),\ldots,(\Delta U_p)(n)).$$

Par théorème 11.5.19, point (6), toutes les $(\Delta U_i)(n)$ appartiennent à $\mathbb{S}^{\vec{N}}_{k+1}$, ce qui prouve que les $U_i(n)$ appartiennent à $\Sigma\mathbb{S}^{\vec{N}}_{k+1}$.

□

Résumons les propriétés de clôtures démontrées dans cette section.

Théorème 13.2.14.
0- Pour tout $U\in\Sigma\mathbb{S}^{\vec{N}}_{k+1}$, $k\geq 1$, et tout entier $c\in\mathbb{N}$,
les suites EU (le décalage de U), $U+\frac{c}{1-X}$ (ajoutant le nombre c à chaque terme), appartiennent à $\Sigma\mathbb{S}^{\vec{N}}_{k+1}$;
si tout $U(n)\geq c$ alors $U-\frac{c}{1-X}$ (soustrayant le nombre c à tout terme) appartient à $\Sigma\mathbb{S}_{k+1}$;
si le nombre $U(0)$ est supérieur ou égal à c, alors la suite définie par $0\mapsto c, n+1\mapsto U(n)$ appartient à $\Sigma\mathbb{S}^{\vec{N}}_{k+1}$.
1- Pour tous $U,V\in\Sigma\mathbb{S}^{\vec{N}}_{k+1}$, $k\geq 1$, la suite $U+V$ appartient à $\Sigma\mathbb{S}^{\vec{N}}_{k+1}$.
2- Pour tout $U,V\in\Sigma\mathbb{S}^{\vec{N}}_{k+1}$, $k\geq 2$, la suite $U\odot V$ (le produit ordinaire) appartient à $\Sigma\mathbb{S}^{\vec{N}}_{k+1}$.
3- Pour toutes suites $U\in\Sigma\mathbb{S}^{\vec{N}}_{k+1}$, et $V\in\Sigma\mathbb{S}_k$, avec $k\geq 2$, la suite $U\times V$ (le produit de convolution) appartient à $\Sigma\mathbb{S}^{\vec{N}}_{k+1}$.
4- Pour tout suite $V\in\Sigma\mathbb{S}_k$, avec $k\geq 2$, telle que $V(0)\geq 1$, la suite U définie par : $U(0)=1$ et $U(n+1)=\sum_{m=0}^{n}U(m)\cdot V(n-m)$ (l'inverse de convolution de $1-XV$) appartient à $\Sigma\mathbb{S}_{k+1}$.
5- Pour toutes suites $U\in\Sigma\mathbb{S}_k$, $V\in\Sigma\mathbb{S}^{\vec{N}}_{\ell}$, la suite $k,l\geq 2$, $U\circ V$ (la composition de suites) appartient à $\Sigma\mathbb{S}^{\vec{N}}_{k+\ell-1}$.
6- Pour tout $k\geq 2$, si $U_1(n),\ldots U_p(n)$ est le vecteur des solutions d'un système d'équations récurrentes exprimées par des polynômes dans $\Sigma\mathbb{S}^{\vec{N}}_{k+1}[X_1,\ldots,X_p]$, avec conditions initiales $U_i(0),U_i(1)\in\mathbb{N}$, avec $U_i(0)\leq U_i(1)$, alors $U_1\in\Sigma\mathbb{S}^{\vec{N}}_{k+1}$.

Rappelons que, par théorème 13.1.3, pour toute suite $U\in\Sigma\mathbb{S}_{k+1}$, le prédicat $P=\{U(n)\mid n\in\mathbb{N}\}$ amène à une structure $\langle\mathbb{N},+1,P\rangle$ ayant une théorie du Second Ordre Monadique décidable.

Chapitre 14

Suites de nombres rationnels

Nous définissons ici une classe de suites de nombres *rationnels* qui peuvent être décrites par automates de niveau k. Les résultats obtenus dans le chapitre 11.5 montrant que de nombreuses opérations sur les suites peuvent être traduites comme des opérations sur les automates sont transposés dans ce cadre plus général.

Définition 14.0.15. *Soit $\mathbb{S}$ un ensemble de suites d'entiers naturels. Nous notons $\mathcal{D}(\mathbb{S})$ l'ensemble des suites $(u_n)_{n\geq 0}$ de la forme :*

$$u_n = a_n - b_n \quad \textit{pour tout } n \geq 0,$$

pour toutes suites $a, b \in \mathbb{S}$. Nous notons $\mathcal{F}(\mathbb{S})$ l'ensemble des suites $(r_n)_{n\geq 0}$ de la forme :

$$r_n = \frac{a_n - b_n}{a'_n - b'_n} \quad \textit{pour tout } n \geq 0,$$

pour toutes suites $a, b, a', b' \in \mathbb{S}$.

Théorème 14.0.16. *Soient u, v des suites de nombres rationnels dans $\mathcal{F}(\mathbb{S}_k)$ (resp. $\mathcal{D}(\mathbb{S}_k)$) pour tout entier $k \geq 3$. Alors les suites de nombres rationnels $u + v, u - v, u \odot v$ sont dans $\mathcal{F}(\mathbb{S}_k)$ (resp. $\mathcal{D}(\mathbb{S}_k)$).*
Si v ne s'annule pas, alors $\frac{u}{v}$ est également dans $\mathcal{F}(\mathbb{S}_k)$.

Preuve : Ce théorème 14.0.16 est évident puisque, par le théorème 11.5.19, $\mathbb{S}_k$ est clos par somme et produit ordinaire.
□

Définissons le problème de l'égalité pour les suites dans $\mathcal{F}(\mathbb{S}_k)$ comme le problème algorithmique suivant :

DONNÉE : deux suites $u, v \in \mathcal{F}(\mathbb{S}_k)$,
QUESTION : $u = v$?

i.e., est il vrai que, $\forall n \in \mathbb{N}, u_n = v_n$?

Corollaire 14.0.17. *Soit $k \geq 3$. Le problème de l'égalité pour les suites dans $\mathcal{F}(\mathbb{S}_k)$ se réduit au problème de l'équivalence pour les automates à k-piles à compteur déterministes.*

Preuve : Notons que $\frac{a-b}{a'-b'} = \frac{c-d}{c'-d'}$ ssi

$$ac' + bd' + a'd + b'c = ad' + bc' + a'c + b'd. \tag{59}$$

Par le théorème 11.5.19, chaque membre de cette équation est reconnu par un seul k-ACD qui peut être construit à partir des huit automates définissant $a, b, c, d, a', b', c', d'$. L'équation (59) peut donc être considérée comme une instance du problème de l'équivalence pour les k-ACD.
□

Notons que, par le théorème 14.0.16, $(\mathcal{D}(\mathbb{S}_k), +, \cdot)$ et $(\mathcal{F}(\mathbb{S}_k), +, \cdot)$ sont des anneaux. Nous denotons par $P(n, X_1, \ldots, X_j, \ldots, X_p)$ tout élément de
$\mathcal{F}(\mathbb{S}_k)[X_1, \ldots, X_j, \ldots, X_p]$ pour mettre en relief le fait que les coefficients de P sont fonctions de l'entier argument n.

Théorème 14.0.18. *Soient $P_i(n, X_1, \ldots, X_j, \ldots, X_p)$, pour $1 \leq i \leq p$, des polynômes à coefficients dans $\mathcal{F}(\mathbb{S}_k)$ ($k \geq 3$) et $c_1, c_2, \ldots, c_p \in \mathbb{Q}$. Soient u_i, pour $1 \leq i \leq p$, les suites définies par*
$u_i(n+1) = P_i(n, u_1(n), \ldots, u_j(n), \ldots u_p(n))$, et $u_i(0) = c_i$. Alors $u_1 \in \mathcal{F}(\mathbb{S}_k)$.

Preuve : Nous supposons que les suites $u_1(n), u_2(n), \ldots, u_p(n)$ vérifient la récurrence

$$u_i(n+1) = P_i(n, u_1(n), \ldots, u_j(n), \ldots u_p(n)) \tag{60}$$

pour $1 \leq i \leq p, n \in \mathbb{N}$ et

$$u_i(0) = c_i. \tag{61}$$

Nous prouvons le théorème en trois étapes.
Étape 1 : Cas où $P_i \in \mathcal{D}(\mathbb{S}_k)[X_1, \ldots, X_p], c_i \in \mathbb{Z}$.
Considérons le polynôme

$$Q_i(n, X_1, Y_1, X_2, Y_2, \ldots, X_p, Y_p) = P_i(n, X_1 - Y_1, X_2 - Y_2, \ldots, X_p - Y_p).$$

Il peut être décomposé comme une somme de monômes de la forme

$$\epsilon \cdot u_n \cdot X_1^{\alpha_1} Y_1^{\beta_1} X_2^{\alpha_2} Y_2^{\beta_2} \cdots X_p^{\alpha_p} Y_p^{\beta_p}$$

où $\epsilon \in \{+1, -1\}, \alpha_i, \beta_i \in \mathbb{N}, (u_n)_{n\in\mathbb{N}} \in \mathbb{S}_k$. Les polynômes Q_i peuvent donc être décomposés ainsi :

$$Q_i(n, X_1, Y_1, X_2, Y_2, \ldots, X_p, Y_p) =$$
$$Q_i^+(n, X_1, Y_1, X_2, Y_2, \ldots, X_p, Y_p) - Q_i^-(n, X_1, Y_1, X_2, Y_2, \ldots, X_p, Y_p)$$

avec $Q_i^+, Q_i^- \in \mathbb{S}_k[X_1, Y_1, X_2, Y_2, \ldots, X_p, Y_p]$. De même,

$$c_i = c_i^+ - c_i^-$$

pour $c_i^+, c_i^- \in \mathbb{N}$. Définissons les nouvelles suites $u_i^+(n), u_i^-(n)$ par :

$$u_i^\epsilon(n+1) = Q_i^\epsilon(n, u_1^+(n), u_1^-(n), \ldots, u_j^+(n), u_j^-(n), \ldots u_p^+(n), u_p^-(n)) \tag{62}$$

pour $1 \leq i \leq p, \epsilon \in \{+, -\}, n \in \mathbb{N}$, et

$$u_i^\epsilon(0) = c_i^\epsilon \tag{63}$$

pour $1 \leq i \leq p, \epsilon \in \{+,-\}$.
À partir de la récurrence (62), des conditions initiales (63) et de la définition de Q_i, Q_i^ϵ, nous pouvons voir que les suites $u_i^+ - u_i^-$ satisfont la récurrence (60) et la condition initiale (61). Donc $u_1 = u_1^+ - u_1^-$ où $u_1^\epsilon \in \mathbb{S}_k$, ce qui montre que $u_1 \in \mathcal{D}(\mathbb{S}_k)$.
Étape 2 : Cas où les $P_i \in \mathcal{F}(\mathbb{S}_k)[X_1, \ldots, X_p]$ sont tous homogènes de même degrés $d \in \mathbb{N}$, $c_i \in \mathbb{Q}$.
Comme l'ensemble $\mathcal{D}(\mathbb{S}_k)$ est clos par produit de Hadamard, nous pouvons supposer que tous les coefficients des polynômes P_i sont des suites de la forme

$$\frac{A(n)}{D(n)}$$

pour différentes suites $A \in \mathcal{D}(\mathbb{S}_k)$ et une seule suite $D \in \mathcal{D}(\mathbb{S}_k)$.
Les équations (60,61) peuvent alors être réécrites :

$$D(n) \cdot u_i(n+1) = R_i(n, u_1(n), \ldots, u_j(n), \ldots u_p(n))$$

pour $1 \leq i \leq p, n \in \mathbb{N}, R_i \in \mathcal{D}(\mathbb{S}_k)[X_1, \ldots, X_p]$, où les R_i sont homogènes de degrés d, et

$$u_i(0) = c_i \forall i \in [1, p].$$

Définissons la suite $F(n)$ par :

$$F(n+1) = D(n) \cdot F(n)^d \text{ pour tout } n \geq 0, \text{ et } F(0) = 1.$$

On peut vérifier que :

$$F(n+1) \cdot u_i(n+1) = R_i(n, F(n)u_1(n), \ldots, F(n)u_j(n), \ldots F(n)u_p(n))$$

pour $1 \leq i \leq p$, $n \in \mathbb{N}$ et

$$F(0) \cdot u_i(0) = c_i.$$

En utilisant l'étape 1 de cette preuve, nous savons que chacune des suites $(F(n) \cdot u_1(n))_{n \geq 0}$ et $(F(n))_{n \geq 0}$ appartiennent à $\mathcal{D}(\mathbb{S}_k)$. Il s'ensuit que $u_1 \in \mathcal{F}(\mathbb{S}_k)$.
Étape 3 : Cas général.
Soit $d \geq 0$ le degrés maximum de tous les polynômes $P_1, \ldots, P_p$. Introduisons une nouvelle indéterminée Z et considérons les polynômes $Q_i(n, X_1, X_2, \ldots, X_p, Z)$ qui sont homogènes de degrés d et tels que

$$Q_i(n, X_1, X_2, \ldots, X_p, 1) = P_i(n, X_1, X_2, \ldots, X_p).$$

Nous introduisons également la suite constante

$$u_{p+1}(n) = 1 \text{ pour tout } n \geq 0.$$

Il est possible de vérifier que les suites $u_1(n), u_2(n), \ldots, u_p(n), u_{p+1}(n)$ vérifient les conditions

$$u_i(n+1) = Q_i(n, u_1(n), \ldots, u_j(n), \ldots u_p(n), u_{p+1}(n))$$

pour $1 \leq i \leq p+1, n \in \mathbb{N}$ et

$$u_i(0) = c_i \text{ si } 1 \leq i \leq p, \quad u_{p+1}(0) = 1.$$

Par l'étape 2 de cette preuve, nous pouvons conclure que $u_1 \in \mathcal{F}(\mathbb{S}_k)$.
□

Théorème 14.0.19. *Soit $u \in \mathcal{F}(\mathbb{S}_{k+1})$ et $v \in \mathcal{F}(\mathbb{S}_k)$ pour un entier $k \geq 3$. Alors le produit de convolution $u \times v$ appartient à $\mathcal{F}(\mathbb{S}_{k+1})$.*

Preuve : Soient $a^+, a^-, b^+, b^- \in \mathbb{S}_{k+1}, c^+, c^-, d^+, d^- \in \mathbb{S}_k$ tels que

$$u = \frac{a^+ - a^-}{b^+ - b^-}, v = \frac{c^+ - c^-}{d^+ - d^-}.$$

Nous introduisons les suites auxiliaires :

$$b = b^+ - b^-, \;\; d = d^+ - d^-; \;\; B(n) = \Pi_{\ell=0}^n b(\ell); \;\; D(n) = \Pi_{\ell=0}^n d(\ell),$$

et

$$\bar{B}(m,n) = \Pi_{\substack{\ell=0 \\ \ell \neq m}}^n b(\ell); \;\; \bar{D}(m,n) = \Pi_{\substack{\ell=0 \\ \ell \neq m}}^n d(\ell).$$

Nous pouvons vérifier que, pour tout $n \in \mathbb{N}$:

$$u \times v(n) = \frac{1}{B(n) \cdot D(n)} \cdot (\sum_{m=0}^{n} a(m)\bar{B}(m,n) \cdot c(n-m)\bar{D}(n-m,n)). \tag{64}$$

Le produit $B(n) \cdot D(n)$ peut être décomposé ainsi :

$$B(n) \cdot D(n) = (B^+(n)D^+(n) + B^-(n)D^-(n)) - (B^+(n)D^-(n) + B^-(n)D^+(n)) \tag{65}$$

où les B^ϵ vérifient les équations

$$\begin{aligned} B_2^\epsilon(n,n) &= b^\epsilon(n) \\ B_2^+(\ell,n) &= b^+(\ell)B_2^+(\ell+1,n) + b^-(\ell)B_2^-(\ell+1,n) \\ B_2^-(\ell,n) &= b^+(\ell)B_2^-(\ell+1,n) + b^-(\ell)B_2^+(\ell+1,n) \end{aligned}$$

pour $0 \leq \ell < n$, et $B^\epsilon(n) = B_2^\epsilon(0,n)$.
Il suit du lemme 12.0.22 que $B_2^\epsilon \in \mathbb{S}_{k+1}^{(2)}$ et du lemme 12.0.24 que $B^\epsilon \in \mathbb{S}_{k+1}$. Similairement, $D^\epsilon \in \mathbb{S}_k$.
La suite double $\bar{B}(m,n)$ peut être définie à travers la suite triple suivante :

$$\bar{B}_3(\ell,m,n) = \Pi_{\substack{\ell'=\ell \\ \ell' \neq m}}^n b(\ell')$$

via la formule

$$\bar{B}(m,n) = \bar{B}_3(0,m,n).$$

Nous pouvons décomposer $\bar{B}_3(\ell,m,n)$ comme

$$\bar{B}_3(\ell,m,n) = \bar{B}_3^+(\ell,m,n) - \bar{B}_3^-(\ell,m,n)$$

où les $\bar{B}_3^\epsilon$ vérifient les équations :

$$\begin{aligned} \bar{B}_3^+(n,n,n) &= 1 \\ \bar{B}_3^-(n,n,n) &= 0 \\ \bar{B}_3^\epsilon(\ell,m,n) &= \bar{B}_3^\epsilon(\ell+1,m,n) \text{ si } \ell = m, \ell < n, \epsilon \in \{+,-\} \\ \bar{B}_3^\epsilon(n,m,n) &= b^\epsilon(n) \text{ si } 0 \leq m < n \\ \bar{B}_3^+(\ell,m,n) &= b^+(\ell)\bar{B}_3^+(\ell+1,m,n) + b^-(\ell)\bar{B}_3^-(\ell+1,m,n) \text{ si } \ell \neq m, \ell < n \\ \bar{B}_3^-(\ell,m,n) &= b^-(\ell)\bar{B}_3^+(\ell+1,m,n) + b^+(\ell)\bar{B}_3^-(\ell+1,m,n) \text{ si } \ell \neq m, \ell < n \end{aligned}$$

Il suit du lemme 12.0.23 que $\bar{B}^\epsilon \in \mathbb{S}_{k+1}^{(2)}$. Similairement, $\bar{D}^\epsilon \in \mathbb{S}_k^{(2)}$.
En utilisant maintenant le lemme 12.0.25 (suites simples vues comme des suites doubles), le lemme 12.0.27 (clôture par produit ordinaire) et le lemme 12.0.28 (clôture par pseudo-convolution), nous obtenons que le numérateur du membre droit de l'équation (64) appartient à $\mathcal{D}(\mathbb{S}_{k+1})$. D'après le théorème 13.2.13, l'ensemble $\mathbb{S}_{k+1}$ est clos par produit ordinaire et somme. La décomposition (65) montre ainsi que le dénominateur du membre droit de l'équation (64) appartient à $\mathcal{D}(\mathbb{S}_{k+1})$. Finalement, $u \in \mathcal{F}(\mathbb{S}_{k+1})$.
□

Comparaison avec d'autres classes L'ensemble des suites *rationnelles* de nombres rationnels est un sous-ensemble de $\mathcal{F}(\mathbb{S}_3)$: de telles suites sont définies par des récurrences de la forme (60,61) où les polynômes P_i sont de degré 1, et les coefficients sont des suites rationnelles constantes. Donc, par le théorème 14.0.18, les suites rationnelles appartiennent à $\mathcal{F}(\mathbb{S}_3)$.
L'ensemble des suites P-*récurrentes* de rationnels est également un sous-ensemble de $\mathcal{F}(\mathbb{S}_3)$: par le théorème 11.4.3, polynômes (avec coefficients dans $\mathbb{N}$) appartiennent à $\mathbb{S}_2$, donc les polynômes (avec coefficients dans $\mathbb{Q}$) appartiennent à $\mathcal{F}(\mathbb{S}_2)$, et par le théorème 14.0.18, les solutions d'équations de polynômes avec coefficients rationnels appartiennent à $\mathcal{F}(\mathbb{S}_3)$.
Rappelons que, l'ensemble des suites P-récurrentes est clos par produit de Hadamard ([Sta80, théorème 2.10]) et également par produit de convolution ([Sta80, théorème 2.3]). Cette dernière propriété n'est pas connue pour l'ensemble $\mathcal{F}(\mathbb{S}_3)$. Le théorème 14.0.19 peut-être vu comme une propriété de clôture "faible" en ce sens.

Perspectives et problèmes ouverts

Différent travaux restent à être effectués pour compléter les résultats présentés ici. Ainsi, nous conjecturons fortement que les graphes de calculs de automates à piles de niveau k admettent une largeur arborescente finie, ce qui généraliserait un résultat connu pour le niveau 1 [Cou90].

Une autre question restant en suspend est de savoir si le *centre* d'un langage de niveau k est de niveau k. Ce résultat est vrai pour $k = 1$ ([BN81]), et à été prouvé ici pour le cas des langages déterministes de niveaux quelconques (proposition 10.1.10). L'étude du cas général passe certainement par l'obtention de propriétés de normalisation des automates permettant de "modérer" les suites de ε-transitions.

Nous avons étudié brièvement le cas des suites doubles, mais il reste à élaborer une vraie théorie sur les suites multiples calculables.

Les études menées dans ce manuscrit font apparaître de nombreuses interrogations.

Nous avons étudié la propriété MD assurant à une structure que toute formule satisfiable admet un modèle définissable. L'arbre complet et la droite infini vérifient cette propriété. A. Rabinovich montre dans un papier non publié [Rab05] que toute extension de la droite des entiers naturels par des relations unaires la vérifie également. Il est prouvé dans [LS98] que la droite de l'ordinal ω^ω ne vérifie pas la propriété *d'uniformisation*. Qu'en est il de la propriété MD qui est une version affaiblie de la propriété d'uniformisation.

Nous avons exhibé des suites finies de relations emboîtées pour lesquelles la droite infinie augmentée de ces relations admet une théorie SOM décidable, comme par exemple la structure $\langle \mathbb{N}, +1, \{n^2\}_{n\in\mathbb{N}}, \{n^4\}_{n\in\mathbb{N}}\rangle$. Toutefois, nous ne savons pas ce qu'il en est pour des relations non emboîtées pour lesquelles, séparément, les logiques sont décidables. Ainsi, nous ne savons pas si la structure $\langle \mathbb{N}, +1, \{n^2\}_{n\in\mathbb{N}}, \{n^3\}_{n\in\mathbb{N}}\rangle$ a une théorie SOM décidable.

Les suites $\mathbb{N}$-rationnelles sont toutes calculables par automates de niveau 2 (proposition 11.4.3), mais nous ne savons pas identifier exactement la classe $\mathbb{S}_2$. Correspond-elle à la classe des suites $\mathbb{N}$-rationnelles ?

Une méthode qui permet de savoir qu'une suite n'est pas calculable par automate est de considérer sa vitesse de croissance. Il est prouvé dans [Dam79] qu'une suite calculée par un automate de niveau k a pour ordre de grandeur $\mathcal{O}(2 \uparrow^k (p(n)))$, où p est un polynôme ($2 \uparrow^0 (n) = n$ et $2 \uparrow^{k+1} (n) = 2^{2\uparrow^k(n)}$). Ceci permet par exemple de montrer que la suite des nombres de Catalan $\frac{C_{2n}^n}{n+1}$ n'est pas calculable par automate. Nous conjecturons qu'une suite k-calculable est de l'ordre de $\mathcal{O}(2 \uparrow^{k-1} (p(n)))$. Ce résultat a été prouvé de façon élégante pour le niveau 2 dans [Gil96].

Nous avons réduit le problème de l'égalité de deux suites de rationnels au problème de l'équivalence de deux automates déterministes ayant des propriétés de normalisation très fortes

puisque leurs graphes forment une droite infinie. Le problème de l'égalité de deux automates déterministe est décidable au niveau 1 ([Sén02]), mais le problème reste ouvert pour les niveaux supérieurs. Au niveau 2, C. Stirling a annoncé récemment la décidabilité du problème pour des automates sans ε-transitions et dont l'ensemble des états est réduit à un unique élément. Nous pouvons donc espérer que le problème est également décidable (au moins au niveau 2), pour nos automates fortement normalisés.

Index des définitions

Index des notations

Bibliographie

[AB88] J.M Autebert and L. Boasson. *Transductions rationnelles. Application aux langages algébriques*. Masson, 1988.

[Aho68] Alfred V. Aho. Indexed grammars—an extension of context-free grammars. *J. Assoc. Comput. Mach.*, 15 :647–671, 1968.

[Aho69] Alfred V. Aho. Nested stack automata. *J. Assoc. Comput. Mach.*, 16 :383–406, 1969.

[Aya73] T. Ayashi. On derivation trees of indexed grammars–An extension of the uvwxy-theorem–. *Publ. RIMS, Kyoto Univ.*, 9 :61–92, 1973.

[Ber79] J. Berstel. *Transductions and context-free languages*. Teubner, 1979.

[Bès01] Alexis Bès. A survey of arithmetical definability. *Bull. Belg. Math. Soc. Simon Stevin*, (suppl.) :1–54, 2001. A tribute to Maurice Boffa.

[BN81] Luc Boasson and Maurice Nivat. Centers of languages. In Peter Deussen, editor, *Theoretical Computer Science*, volume 104 of *Lecture Notes in Comput. Science*, pages 245–251. Springer, 1981.

[BR88] J. Berstel and C. Reutenauer. *Rational series and their languages*. Springer-Verlag, 1988.

[Büc60] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6 :66–92, 1960.

[Büc62] J. Richard Büchi. On a decision method in restricted second order arithmetic. In *Logic, Methodology and Philosophy of Science (Proc. 1960 Internat. Congr .)*, pages 1–11. Stanford Univ. Press, Stanford, Calif., 1962.

[Cac03] Thierry Cachat. Higher order pushdown automata, the Caucal hierarchy of graphs and parity games. In *Automata, languages and programming*, volume 2719 of *Lecture Notes in Comput. Sci.*, pages 556–569. Springer, Berlin, 2003.

[Car05] Arnaud Carayol. Regular sets of higher-order pushdown stacks. In *MFCS*, volume 3618 of *Lecture Notes in Comput. Sci.*, pages 168–179. Springer, 2005.

[Cau96] Didier Caucal. On infinite transition graphs having a decidable monadic theory. In *Automata, languages and programming (Paderborn, 1996)*, volume 1099 of *Lecture Notes in Comput. Sci.*, pages 194–205. Springer, Berlin, 1996.

[Cou83] Bruno Courcelle. Fundamental properties of infinite trees. *Theoret. Comput. Sci.*, 25(2) :95–169, 1983.

[Cou90] Bruno Courcelle. Graph rewriting : an algebraic and logic approach. In *Handbook of theoretical computer science, Vol. B*, pages 193–242. Elsevier, Amsterdam, 1990.

[Cou95] Bruno Courcelle. The monadic second-order logic of graphs. IX. Machines and their behaviours. *Theoret. Comput. Sci.*, 151(1) :125–162, 1995. Topology and completion in semantics (Chartres, 1993).

[CT02] Olivier Carton and Wolfgang Thomas. The monadic theory of morphic infinite words and generalizations. *Inform. and Comput.*, 176(1) :51–65, 2002.

[CW98] Bruno Courcelle and Igor Walukiewicz. Monadic second-order logic, graph coverings and unfoldings of transition systems. *Ann. Pure Appl. Logic*, 92(1) :35–62, 1998.

[CW03] A. Carayol and S. Wöhrle. The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In *FST TCS 2003 : Foundations of software technology and theoretical computer science*, volume 2914 of *Lecture Notes in Comput. Sci.*, pages 112–123. Springer, Berlin, 2003.

[Dam79] Werner Damm. An algebraic extension of the chomsky-hierarchy. In Jirí Becvár, editor, *MFCS*, volume 74 of *Lecture Notes in Comput. Sci.*, pages 266–276. Springer, 1979.

[Dam82] Werner Damm. The IO- and OI-hierarchies. *Theoret. Comput. Sci.*, 20(2) :95–207, 1982.

[DG86] Werner Damm and Andreas Goerdt. An automata-theoretical characterization of the OI-hierarchy. *Inform. and Control*, 71(1-2) :1–32, 1986.

[EJ91] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings of the 32nd annual symposium on Foundations of computer science*, pages 368–377, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[Eng83] J. Engelfriet. Iterated pushdown automata and complexity classes. In *Proceedings of the 14th Symposium on Theory of Computing*, pages 365–373. Association for Computing Machinery, 1983.

[Eng91] J. Engelfriet. Iterated stack automata and complexity classes. *Inform. and Comput.*, 95(1) :21–75, 1991.

[ER66] Calvin C. Elgot and Michael O. Rabin. Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. *J. Symbolic Logic*, 31(2) :169–181, 1966.

[ES84] Joost Engelfriet and Giora Slutzki. Extended macro grammars and stack controlled machines. *J. Comput. System Sci.*, 29(3) :366–408, 1984.

[EV86] Joost Engelfriet and Heiko Vogler. Corrigenda : "Pushdown machines for the macro tree transducer". *Theoret. Comput. Sci.*, 48(2-3) :339 (1987), 1986.

[Fis68] M.J. Fischer. *Grammars with macro-like productions.* PhD thesis, Harvard University, 1968. See also :Proc. 9th Symp. on Switching and Automata Theory (1968) p. 131-142.

[FS03] S. Fratani and G. Sénizergues. Iterated pushdown automata and sequences of rational numbers. Presented to *Second St.Petersburg Days of Logic and Computability, 2003.* To appear in APAL, available version at http ://dept-info.labri.u-bordeaux.fr/~fratani, 2003.

[GH82] Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proceedings of the 14th Symposium on Theory of Computing*, pages 60–65. Association for Computing Machinery, 1982.

[Gil96] Robert H. Gilman. A shrinking lemma for indexed languages. *Theoret. Comput. Sci.*, 163(1-2) :277–281, 1996.

[Gre67] Sheila A. Greibach. A note on pushdown store automata and regular systems. *Proc. Amer. Math. Soc.*, 18 :263–268, 1967.

[Gre70] Sheila A. Greibach. Full AFLs and nested iterated substitution. *Information and Control*, 16 :7–35, 1970.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games : A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[Han77] *Handbook of mathematical logic*, chapter Decidable theories. M.O.Rabin, pages 595–629. North-Holland Publishing Co., Amsterdam, 1977. Edited by Jon Barwise, With the cooperation of H. J. Keisler, K. Kunen, Y. N. Moschovakis and A. S. Troelstra, Studies in Logic and the Foundations of Mathematics, Vol. 90.

[Har78] M.A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, Reading, Mass., 1978.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Co., Reading, Mass., 1979. Addison-Wesley Series in Computer Science.

[KNU02] Teodor Knapik, Damian Niwiński, and Paweł Urzyczyn. Higher-order pushdown trees are easy. In *Foundations of software science and computation structures (Grenoble, 2002)*, volume 2303 of *Lecture Notes in Comput. Sci.*, pages 205–222. Springer, Berlin, 2002.

[LS98] Shmuel Lifsches and Saharon Shelah. Uniformization and Skolem functions in the class of trees. *J. Symbolic Logic*, 63(1) :103–127, 1998.

[Mae99] Arnaud Maes. An automata-theoretic decidability proof for first-order theory of $\langle \mathbf{N}, <, P \rangle$ with morphic predicate P. *J. Autom. Lang. Comb.*, 4(3) :229–245, 1999. Journées Montoises d'Informatique Théorique (Mons, 1998).

[Mas74] A. N. Maslov. The hierarchy of index languages of arbitrary level. *Dokl. Akad. Nauk SSSR*, 217 :1013–1016, 1974.

[Mas76] A. N. Maslov. Multilevel pushdown automata. *Problemy Peredači Informacii*, 12(1) :55–62, 1976.

[MS92] A. Muchnik and A.L. Semenov. Automata on infinite objects, monadic theories, and complexity. In *Dagstuhl-Seminar*, 1992.

[Niv68] Maurice Nivat. Transductions des langages de Chomsky. *Ann. Inst. Fourier (Grenoble)*, 18(fasc. 1) :339–455, 1968.

[PWZ96] M. Petkovšek, H.S. Wilf, and D. Zeilberger. $A = B$. A. K. Peters Ltd., Wellesley, MA, 1996.

[Rab69] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141 :1–35, 1969.

[Rab05] A. Rabinovich. On decidability of Monadic logic of order over the naturals extended by monadic predicates. 2005.

[Sem84] A. L. Semenov. Decidability of monadic theories. In *Mathematical foundations of computer science, 1984 (Prague, 1984)*, volume 176 of *Lecture Notes in Comput. Sci.*, pages 162–175. Springer, Berlin, 1984.

[Sén02] Géraud Sénizergues. $L(A) = L(B)$? A simplified decidability proof. *Theoret. Comput. Sci.*, 281(1-2) :555–608, 2002. Selected papers in honour of Maurice Nivat.

[Sie70] D. Siefkes. Decidable extensions of monadic second order successor arithmetic. In *Automatentheorie und formale Sprachen (Tagung, Math. Forschungsinst., Oberwolfach, 1969)*, pages 441–472. Bibliographisches Inst., Mannheim, 1970.

[Sta80] R.P. Stanley. Differentiably finite power series. *European Journal of Combinatorics 1*, pages 175–188, 1980.

[Tho78] Wolfgang Thomas. The theory of successor with an extra predicate. *Math. Ann.*, 237(2) :121–132, 1978.

[Tho90] Wolfgang Thomas. Automata on infinite objects. In *Handbook of theoretical computer science, Vol. B*, pages 133–191. Elsevier, Amsterdam, 1990.

[Tho03] Wolfgang Thomas. Constructing infinite graphs with a decidable MSO-theory. In *Mathematical foundations of computer science 2003*, volume 2747 of *Lecture Notes in Comput. Sci.*, pages 113–124. Springer, Berlin, 2003.

[Wal96] Igor Walukiewicz. Monadic second order logic on tree-like structures. In *STACS 96 (Grenoble, 1996)*, volume 1046 of *Lecture Notes in Comput. Sci.*, pages 401–413. Springer, Berlin, 1996.

[Wal01] Igor Walukiewicz. Pushdown processes : games and model-checking. *Inform. and Comput.*, 164(2) :234–263, 2001. FLOC '96 (New Brunswick, NJ).

[Wal02] I. Walukiewicz. Monadic second-order logic on tree-like structures. *Theoret. Comput. Sci.*, 275(1-2) :311–346, 2002.