

UNIVERSITÉ — — PARIS-EST

THÈSE

Pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ PARIS-EST

Dynamique symbolique des systèmes 2D et des arbres infinis

Spécialité **Informatique**
École doctorale MSTIC

Soutenue publiquement par **Nathalie AUBRUN**
le mercredi **22 juin 2011**

COMPOSITION DU JURY :

Mme. Marie-Pierre BÉAL	directrice
M. Jérôme BUZZI	examinateur
M. Cătălin DIMA	examinateur
Mme. Christiane FROUGNY	rapporteur
M. Michael HOCHMAN	rapporteur
M. Jarkko KARI	rapporteur
M. Mathieu SABLİK	co-directeur
M. Alexander SHEN	examinateur

Résumé

Cette thèse est consacrée à l'étude des décalages, ou systèmes dynamiques symboliques, définis sur certains monoïdes finiment présentés : \mathbb{Z}^d d'une part et les arbres d'autre part. Le principal résultat concernant les décalages multidimensionnels établit que tout décalage effectif de dimension d est obtenu par facteur et sous-action projective d'un décalage de type fini de dimension $d + 1$. De ce résultat on peut tirer de nombreuses applications, en particulier le fait que les décalages S-adiques multidimensionnels donnés par une suite effective de substitutions sont sofiques. Sur les décalages d'arbres nous montrons un théorème de décomposition, duquel nous déduisons la décidabilité du problème de conjugaison entre deux décalages d'arbres de type fini. Nous nous intéressons ensuite à la classe des décalages d'arbres sofiques, qui sont exactement ceux reconnus par des automates d'arbres. Nous montrons que tout décalage d'arbres sofique possède un unique automate d'arbres déterministe, réduit, irréductible et synchronisé qui le reconnaît. Enfin nous montrons que l'appartenance à la sous-classe des décalages d'arbres AFT est décidable.

Mots-clé

dynamique symbolique, systèmes dynamiques symboliques multidimensionnels, décalages effectifs, sous-action projective, décalages S-adiques, automates d'arbres, décalages d'arbres

Abstract

This thesis is devoted to the study of subshifts, or symbolic dynamical systems, defined on some finitely presented monoids like \mathbb{Z}^d or the infinite binary tree. The main result concerning multidimensional subshifts establishes that any effective subshift of dimension d can be obtained by factor map and projective subaction of a subshift of finite type of dimension $d + 1$. This result has many applications, and in particular we prove that multidimensional effective S-adic subshifts are sofic. On tree-shifts we prove a decomposition theorem, which implies that the conjugacy problem between two tree-shifts of finite type is decidable. We then investigate the class of sofic tree-shifts that are exactly those recognized by tree automata. We prove that any sofic tree-shift has a unique deterministic, reduced, irreducible and synchronized tree automaton that recognized it. Finally we prove that it is decidable whether a sofic tree-shift belong to the sub-class of AFT tree-shifts.

Keywords

symbolic dynamics, multidimensional symbolic dynamical systems, effective subshifts, projective subaction, S-adic subshifts, tree automata, tree-shifts

Table des matières

Introduction	1
Motivation	6
Résultats principaux	6
1 Dynamique symbolique sur un monoïde	9
1.1 Monoïdes finiment présentés	10
1.1.1 Définition	10
1.1.2 Graphe de Cayley	11
1.1.3 Cas des groupes finiment présentés	12
1.2 Les décalages, deux points de vue complémentaires	13
1.2.1 Point de vue topologique	13
1.2.2 Définition par motifs interdits	15
1.2.3 Coïncidence des définitions	19
1.3 Classes de décalages	20
1.3.1 Décalages de type fini	20
1.3.2 Décalages sofiqes	21
1.3.3 Décalages effectifs	23
2 Codage d'une machine de Turing dans un décalage sur \mathbb{Z}^2	29
2.1 Décalages substitutifs et décalages S-adiques	30
2.1.1 Substitutions	30
2.1.2 Composition de substitutions	33
2.1.3 Décalages S-adiques	33
2.1.4 Décalages substitutifs non déterministes	34
2.1.5 Substitutions non-déterministes et théorème de Mozes	36
2.2 Opérations et simulation	38
2.2.1 Simulation et classes stables	39
2.2.2 Exemples d'opérations et résultats classiques de simulation	39
2.2.3 La sous-action projective	41
2.3 Décalage sofique codant une machine de Turing	46
2.3.1 Machines de Turing dans un SFT	46
2.3.2 Espaces de calcul pour MT	50

2.3.3	Communication dans le décalage $\mathbf{T}_{\{s_1\}}$	53
2.3.4	Initialisation des calculs	55
2.3.5	Diagramme espace-temps d'une MT dans un décalage so- fique	57
3	Simulation de décalages effectifs 1D par des décalages de type fini 2D	62
3.1	Un résultat de simulation	63
3.1.1	Énoncé du résultat et idée de la preuve	63
3.1.2	Communication entre machines	65
3.1.3	Génération de motifs interdits	70
3.1.4	Détection de motifs interdits	72
3.1.5	Construction finale	76
3.2	Deux applications du théorème de simulation	77
3.2.1	Ordres sur les langages de motifs et son équivalent sur les décalages	78
3.2.2	Décalages S-adiques multidimensionnels effectifs	86
4	Décalages d'arbres	93
4.1	Généralités	94
4.1.1	Automate d'arbres	94
4.1.2	Décalages d'arbres irréductibles	99
4.1.3	Contexte, automate réduit et bloc synchronisant	101
4.1.4	Automates minimaux	103
4.1.5	Théorème de décomposition	108
4.2	Décalages de type fini	111
4.2.1	Décalages de sommets et de transitions	111
4.2.2	Conjugaison des décalages de type fini	112
4.3	Les décalages presque de type fini (AFT)	116
4.3.1	Définition	116
4.3.2	Caractérisation des AFT par leur couverture de Shannon	119
4.4	Procédures de décision	120
4.4.1	Calcul de la couverture de Shannon	120
4.4.2	Automate des paires et graphe des paires	121
4.4.3	Localité	125
4.4.4	Automate fermant à gauche	126
	Conclusion et perspectives	128
	Conclusion	128
	Théorème de décomposition général	129

Point de vue algébrique	130
Table des figures	130
Bibliographie	137

Remerciements

Mes premiers remerciements vont bien évidemment à mes deux directeurs, Marie-Pierre et Mathieu. Pendant ces trois années ils ont tous deux été particulièrement disponibles, et m'ont permis de préparer cette thèse dans les meilleures conditions. Marie-Pierre merci d'avoir accepté d'encadrer ma thèse sans même me connaître et d'avoir toujours trouvé du temps à me consacrer. Mathieu merci de m'avoir aiguillé vers ce domaine de recherche, merci de ton accueil lors de mes différents séjours à Marseille, d'avoir toujours eu des questions pour me relancer lorsque c'était nécessaire. Merci à tous les deux de m'avoir fait part de votre vision du monde de la recherche, et pour tous les conseils que vous avez pu me donner.

Je tiens ensuite à remercier les membres de mon jury. Merci à Christiane Frougny d'avoir accepté de relire ma thèse. Haluaisin kiittää Jarkko Karia ranskankielisen väitöskirjani lukemisesta ja kutsumisesta tutkimusryhmäänsä ensi vuonna. I would like to warmly thank Mike Hochman for having accepted to review my thesis. Merci à Jérôme Buzzi d'avoir accepté de faire partie de mon jury, et d'en endosser le rôle de président. Я также выражаю благодарность Александру Шеңу за его участие в работе диссертационной комиссии. Merci également à Cătălin Dima d'avoir accepté de faire partie de mon jury.

Merci à tous les chercheurs et étudiants que j'ai pu cotoyer pendant ces trois années, pendant les rencontres de l'ANR Subtile, celles du groupe de travail Pythéas Fogg, ou à l'occasion de conférences et séminaires auxquels j'ai assisté.

Je remercie aussi les membres du LIGM, et en particulier les secrétaires du laboratoire, pour leur efficacité et leur bonne humeur.

Merci à mes amis pour les bons moments partagés pendant cette période, que ce soit pour une soirée, un week-end, des vacances, un entraînement de tennis ou un match de foot.

Merci à toute ma famille, au sens le plus large, que j'ai toujours autant de plaisir à retrouver et sur laquelle je sais que je peux compter.

Bastien, merci d'avoir été à mes côtés pendant tout ce temps, merci de t'être toujours intéressé à mon travail, et enfin merci pour ton soutien lorsque j'en ai eu besoin.

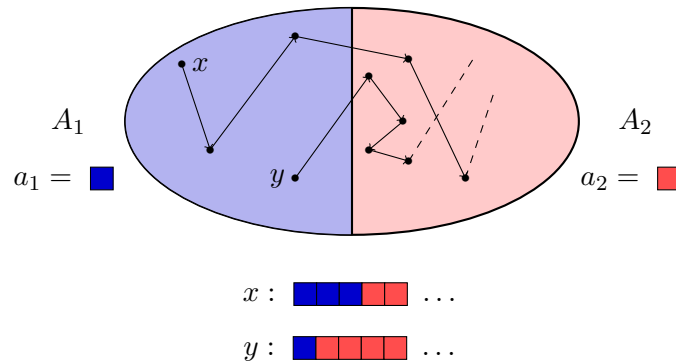
Et enfin Benjamin et sa future petite soeur, sans qui tout cela n'aurait pas été assez difficile. . .

Introduction

Cette thèse est dévolue à l'étude des décalages, qui sont des objets mathématiques utilisés pour modéliser les systèmes dynamiques à temps discret. Un système dynamique discret (X, F) consiste en un ensemble de configurations X compact (appelé l'espace des phases) dont l'évolution au cours du temps est donnée par une fonction $F : X \rightarrow X$ continue et compacte. Chaque configuration initiale $x_0 \in X$ décrit une trajectoire $(x_n)_{n \in \mathbb{N}}$ dans l'espace, où $x_n = F^n(x_0)$ pour tout $n \in \mathbb{N}$. Pour étudier ces trajectoires, il est pertinent pour simplifier le problème de discrétiser les états du système dynamique en se donnant une partition $X = \bigcup_{i=1}^d A_i$ de l'espace des phases. En notant A l'alphabet fini $\{a_1, \dots, a_n\}$, on s'intéresse alors à l'ensemble

$$\mathbf{T} = \overline{\{(x_n) \in A^{\mathbb{N}} \mid \exists x \in X, \forall n \in \mathbb{N}, F^n(x) \in A_i \Leftrightarrow x_n = a_i\}},$$

qui est l'adhérence de l'ensemble des suites d'éléments de la partition choisie par lesquels passe l'orbite d'un point de l'espace des phases [HM38].

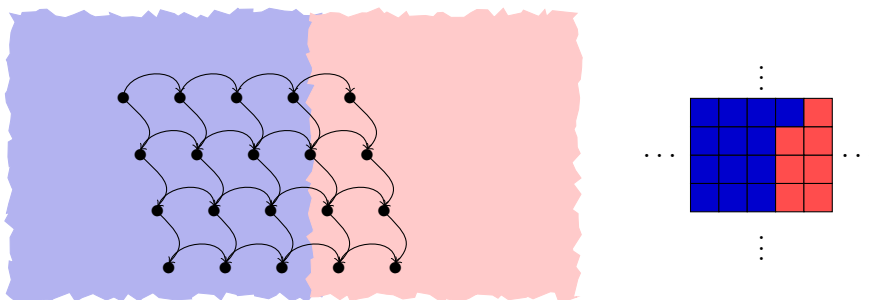


De manière plus générale, si \mathbb{M} est un monoïde et F une \mathbb{M} -action sur l'espace des phases X , on peut modéliser le système (X, F) en se donnant une partition de X , et de manière analogue on définit l'ensemble

$$\mathbf{T} = \overline{\{(x_m) \in A^{\mathbb{M}} \mid \exists x \in X, \forall m \in \mathbb{M}, F^m(x) \in A_i \Leftrightarrow x_m = a_i\}},$$

que l'on appelle décalage associé au système dynamique (X, F) et à la partition $(A_i)_{i=1\dots d}$.

$$A = \{ \blacksquare, \blacksquare \}$$



Étant donné un alphabet fini A et un monoïde \mathbb{M} , un décalage est donc un ensemble de configurations $\mathbf{T} \subseteq A^{\mathbb{M}}$ qui respecte les conditions suivantes : l'ensemble \mathbf{T} est un ensemble fermé, et l'ensemble \mathbf{T} est stable par translation, qui est l'action naturelle $\sigma_{\mathbb{M}}$ de \mathbb{M} sur $A^{\mathbb{M}}$: $\sigma_{\mathbb{M}}^i(\mathbf{T}) \subseteq \mathbf{T}$ pour tout élément i du monoïde \mathbb{M} . Le comportement du système dynamique apparaît ainsi dans le décalage associé avec une certaine imprécision, due à la pertinence de la partition choisie. Un décalage est donc une approximation discrète d'un système dynamique, et cette définition des décalages suffit à justifier leur étude.

On peut aussi adopter une approche combinatoire de l'objet décalage, dont nous venons de voir la définition topologique. Retenons simplement pour l'instant qu'un décalage est un ensemble de configurations $\mathbf{T} \subseteq A^{\mathbb{M}}$ qui vérifient certaines conditions, ou règles locales, ces conditions codant le comportement d'un système dynamique vu au travers de l'ensemble A . Sur le premier exemple, la seule condition pour qu'une suite bi-infinie de \blacksquare et de \blacksquare appartienne au décalage \mathbf{T} est qu'un symbole \blacksquare est toujours suivi d'un autre symbole \blacksquare . Cette contrainte peut s'exprimer de la manière suivante : on interdit la présence du motif $\blacksquare\blacksquare$ dans les configurations $x \in \{ \blacksquare, \blacksquare \}^{\mathbb{N}}$ du décalage \mathbf{T} . Étant donné un ensemble de motifs finis sur un monoïde \mathbb{M} , on peut définir un ensemble de configurations, appelé \mathbf{T}_F , qui regroupe les configurations dans lesquelles aucun motif de l'ensemble F n'apparaît :

$$\mathbf{T}_F = \{ x \in A^{\mathbb{M}} \mid \forall m \in F, m \text{ n'apparaît pas dans } x \}.$$

Un résultat classique sur \mathbb{Z} , facilement adaptable à n'importe quel monoïde \mathbb{M} , montre que les ensembles de configurations ainsi définis par motifs interdits

sont exactement les décalages au sens topologique. Enfin plutôt que les motifs interdits, on peut s'intéresser aux motifs autorisés dans un décalage, qui sont les motifs qui apparaissent dans au moins une configuration. L'ensemble de tous ces motifs forme le langage du décalage, qui suffit à le définir. Cette approche possède l'avantage de donner la liste exhaustive des motifs autorisés ; en contrepartie un tel ensemble de motifs doit répondre à certaines exigences (être prolongeable et stable par sous-motif), ce qui le rend difficile à exploiter. La définition par exclusion de motifs est donc plus aisée à utiliser en pratique. Il est à noter qu'un même décalage peut être défini par une infinité d'ensembles de motifs interdits, et que le plus grand d'entre eux au sens de l'inclusion est le complémentaire du langage. On peut ainsi définir différentes classes de décalages en fonction des propriétés vérifiées par au moins l'un de ses ensembles de motifs interdits.

La première classe, la plus simple pour la définition par exclusion de motifs, est la classe des décalages de type fini. Un décalage appartient à cette classe si on peut trouver un ensemble fini de motifs interdits qui le définit. Sur \mathbb{N} ou \mathbb{Z} , les décalages de type fini possèdent des applications en théorie des codes [Mar85], et leur étude est grandement facilitée par l'existence d'une représentation par un graphe [Fis75]. En particulier, savoir si un décalage de type fini de dimension 1 est vide est un problème décidable, équivalent à l'existence d'une configuration périodique dans le décalage. Mais cette bonne connaissance des décalages de type fini se dégrade dès que l'on augmente la dimension. D'une part, il n'existe pas de représentation aussi simple qu'en dimension 1 ; des représentations des décalages de type fini sur \mathbb{Z}^2 existent bien, par exemple les systèmes textiles [Nas95], cependant elles sont difficilement exploitables en raison de leur caractère extrêmement technique. D'autre part un décalage de type fini sur \mathbb{Z}^2 est toujours conjugué à un ensemble de pavages défini par des tuiles de Wang [Wan61], et les résultats de la théorie des pavages se transmettent donc aux décalages de type fini. En particulier les problèmes précédemment cités deviennent indécidables [Ber66, Rob71]. Le point clé de ces preuves réside dans la construction d'un décalage de type fini ne contenant que des configurations aperiodiques (c'est-à-dire qui ne sont laissées invariantes par aucune translation).

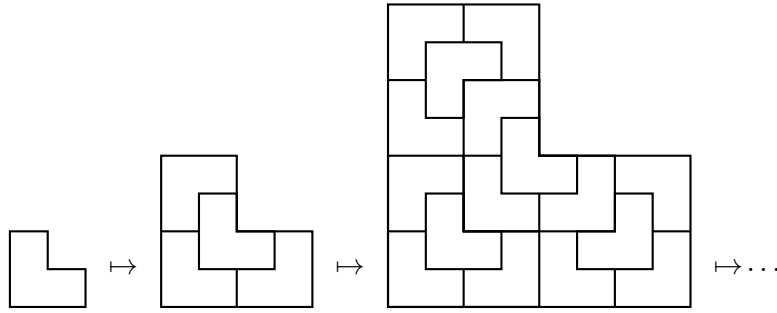
La classe des décalages sofiques, initialement introduite par Weiss [Wei73] pour la dimension 1, est très proche de la classe des décalages de type fini. Un décalage sofique y est défini comme l'image d'un décalage de type fini par une transformation locale sur les lettres de l'alphabet. Cette transformation est opérée par une fonction de bloc, définie sur l'espace des configurations, qui peut

être vue de manière équivalente soit comme une application continue et commutant avec la translation, soit comme une application globale définie par une fonction locale [Hed69]. En dimension 1, un décalage sofique peut toujours être reconnu par un graphe fini, les configurations du décalage étant les étiquettes de chemins infinis ou bi-infinis dans ce graphe [Fis75]. Le langage d'un décalage sofique est donc un langage rationnel, c'est-à-dire accepté par un automate fini. On peut étendre la définition des décalages sofiques sur n'importe quel monoïde, mais savoir si un décalage est sofique ou non devient un problème difficile. On sait néanmoins qu'un décalage n'est pas sofique s'il est minimal et d'entropie positive, ou bien si pour toute configuration du décalage, la complexité de Kolmogorov de chaque motif de taille n est plus que linéaire [DLS01]. Citons aussi l'exemple du décalage impair, ou *odd shift*, qui est sofique en dimensions 1 et 2 [CH], alors que le décalage pair, ou *even shift*, est sofique en dimension 1 mais pas en dimension 2.

Enfin la troisième classe à laquelle nous nous intéresserons dans cette thèse est la classe des décalages effectifs. Ce sont les décalages qui peuvent être définis par un ensemble de motifs interdits pouvant être énumérés par un algorithme, ou de manière équivalente par une machine de Turing. Il s'agit donc de la classe des décalages qui sont calculables, et elle apparaît naturellement lorsque l'on s'intéresse aux sous-dynamiques des décalages sofiques ou de type fini [Hoc09].

Une question fondamentale en dynamique symbolique est celle de la conjugaison des décalages. Deux décalages définis sur un même monoïde sont conjugués s'il existe une fonction de bloc bijective qui transforme l'un des décalages en l'autre. Deux décalages conjugués sont donc identiques à un recodage près. La décidabilité de la conjugaison des décalages de type fini sur \mathbb{Z} est actuellement une question ouverte. On sait que sur \mathbb{N} , la conjugaison des sofiques est un problème décidable [Wil73], mais que le passage à la dimension supérieure rend le même problème indécidable même pour les décalages de type fini.

Un autre exemple qui illustre comment la dimension influe sur les propriétés des décalages est celui des décalages substitutifs. Un moyen géométrique pour construire des pavages du plan est d'utiliser des substitutions, qui permettent notamment d'obtenir des pavages apériodiques. Étant donné un ensemble fini de tuiles τ , une substitution s est une fonction qui agit comme une similitude de sorte que, pour toute tuile $t \in \tau$, l'image $s(t)$ est une union finie de copies de certaines des tuiles dans τ . La substitution s agit donc en deux temps sur une tuile t : d'abord la tuile est *gonflée*, puis elle est divisée en nouvelles tuiles.



En itérant, si cela est possible, la substitution sur une des tuiles, on obtient ainsi un pavage du plan. Dans le cadre de cette thèse nous nous restreignons aux substitutions rectangulaires, qui sont donc données par des règles de la forme

$$a \mapsto \begin{array}{|c|c|c|} \hline a_{1,l} & \dots & a_{k,l} \\ \hline \vdots & & \vdots \\ \hline a_{1,1} & \dots & a_{k,1} \\ \hline \end{array}.$$

Cela permet de définir les décalages substitutifs comme les décalages dont les configurations sont celles qui peuvent être dé-substituées un nombre arbitrairement grand de fois. Les trois classes de décalages (de type fini, sofiques et effectifs) que nous avons présentées précédemment correspondent à différentes complexités (au sens de la hiérarchie de Chomsky pour les langages) du langage d'un décalage. A quel niveau se situent les décalages substitutifs dans cette hiérarchie? Pour toute substitution, il est possible de calculer de manière effective la suite des itérées de la substitution sur chacune des lettres de l'alphabet; un décalage substitutif est donc effectif. Cependant en dimension 1 certains décalages substitutifs sont sturmiens [Dur00], et un argument combinatoire (impliquant la notion de complexité de facteur [HM38] et le théorème de Pansiot [Pan84]) montre que ces décalages ne peuvent pas être sofiques, car les seuls décalages sofiques de dimension 1 qui sont aussi substitutifs sont périodiques. Ainsi les décalages substitutifs et les décalages sofiques forment deux classes distinctes. En dimension supérieure, sur \mathbb{Z}^d , la situation est complètement différente puisque la classe des décalages substitutifs et celle des décalages sofiques coïncident (voir [Moz89] pour le cas des substitutions rectangulaires et [GS98] pour des substitutions plus générales). Une nouvelle fois le passage à la dimension supérieure induit un changement de comportement.

Motivation

Les décalages de dimension 1 sur \mathbb{N} ou \mathbb{Z} sont des objets très bien compris, mais le passage à la dimension supérieure rend des questions fondamentales sur les décalages indécidables comme nous l'avons vu sur quelques exemples, et les résultats valides en dimension 1 ne le sont plus nécessairement en dimension supérieure. Savoir si un décalage contient au moins une configuration est ainsi indécidable sur \mathbb{N}^d ou \mathbb{Z}^d dès que $d \geq 2$. Le problème de la conjugaison des décalages, qui est une des questions centrales de la dynamique symbolique, est décidable sur \mathbb{N} , indécidable sur \mathbb{Z}^2 et constitue une question ouverte sur \mathbb{Z} . Ainsi la structure sur laquelle sont définis les décalages conditionne directement la complexité de ces problèmes. Cette thèse est motivée par ces observations.

Dans une première partie, qui correspond aux Chapitres 2 et 3, nous nous intéressons aux décalages sur \mathbb{Z}^d avec $d \geq 2$. Ces décalages multidimensionnels sont encore mal compris, mais de récents travaux permettent de mieux comprendre ce que sont les décalages de type fini sur \mathbb{Z}^d , par exemple en caractérisant les valeurs possibles de leur entropie [HM10] ou en étudiant leurs dynamiques directionnelles [Hoc09]. Nous nous inspirerons de ces travaux afin de faire apparaître des liens entre les différentes classes de décalages précédemment définies.

Une tentative pour comprendre où se situe fondamentalement la différence entre décalages sur \mathbb{N} et sur \mathbb{N}^2 est d'étudier les décalages définis sur une structure intermédiaire, par exemple le monoïde libre à deux générateurs \mathbb{M}_2 . Dans cette optique nous nous intéressons aux décalages d'arbres dans une seconde partie, qui correspond au Chapitre 4. Les automates finis d'arbres sont des objets déjà étudiés [CDG⁺07] et qui présentent de fortes similitudes avec les automates sur les mots. L'objectif de cette approche est de comprendre à quel point les décalages d'arbres sont proches des décalages sur \mathbb{N} ou sur \mathbb{N}^2 , en s'intéressant notamment au problème de conjugaison.

Résultats principaux

Le premier chapitre est dédié aux définitions et résultats généraux pour des décalages sur un monoïde, en se restreignant au cas des monoïdes finiment présentés. Le choix de travailler sur de tels monoïdes est motivé d'une part par la simplicité de leur représentation, puisqu'on peut les décrire de manière finie, et d'autre part par la complexité suffisante de leur structure, puisqu'en

particulier le problème du mot peut y être indécidable [Col86].

Le Chapitre 2 est consacré à la construction d'un décalage sofique sur \mathbb{Z}^2 permettant de coder le comportement d'une machine de Turing, à l'aide notamment d'un décalage substitutif ne possédant que des configurations apériodiques. Le décalage sofique dans lequel apparaissent les diagrammes espace-temps de machine de Turing est construit à l'aide d'opérations élémentaires sur les décalages. Cette construction est celle présentée dans [AS10].

Le Chapitre 3 présente un résultat de simulation qui permet de faire le lien, via les opérations de facteur et de sous-action projective, entre la classe des décalage de type fini sur \mathbb{Z}^{d+1} et la classe des décalages effectifs sur \mathbb{Z}^d . Nous montrons que tout décalage effectif de dimension d peut être obtenu comme facteur et sous-action projective d'un décalage de type fini de dimension $d + 1$ [AS10]. Ce résultat améliore celui d'Hochman [Hoc09] dont la construction nécessite un décalage de type fini de dimension $d + 2$. Une preuve différente, basée sur des pavages auto-similaires, a été obtenue dans le même temps [DRS10b]. Nous utiliserons ce résultat pour montrer que les décalages S-adiques multidimensionnels définis par une suite effective de substitutions sont sofiques [AS11].

La dernière partie, qui correspond au Chapitre 4, est consacrée à l'étude des décalages d'arbres, qui avec le formalisme du Chapitre 1 sont les décalages sur le monoïde libre à deux générateurs \mathbb{M}_2 . Plus précisément, nous nous intéresserons à la classe des décalages d'arbres sofiques pour lesquels il existe une bonne notion d'automate, conduisant à des résultats très similaires à ceux déjà connus en dimension 1 sur \mathbb{N} ou \mathbb{Z} . Dans un premier temps nous montrerons que la conjugaison des décalages d'arbres de type fini est un problème décidable [AB09], généralisant ainsi le résultat de Williams pour des décalages sur \mathbb{N} [Wil73]. Nous définirons ensuite une notion de décalage d'arbres presque de type fini (AFT), qui étend celle connue pour la dimension 1 [Mar85], et montrerons que l'appartenance d'un décalage d'arbres sofique à cette classe est décidable [AB10].

Enfin dans la conclusion nous proposerons quelques pistes pour donner suite aux résultats présentés dans cette thèse.

1 Dynamique symbolique sur un monoïde finiment présenté

1.1	Monoïdes finiment présentés	10
1.1.1	Définition	10
1.1.2	Graphe de Cayley	11
1.1.3	Cas des groupes finiment présentés	12
1.2	Les décalages, deux points de vue complémentaires	13
1.2.1	Point de vue topologique	13
	L'espace des configurations	13
	Définition topologique des décalages	14
1.2.2	Définition par motifs interdits	15
	Motifs et langages	15
	Motifs interdits et langage d'un décalage sur M	16
1.2.3	Coïncidence des définitions	19
1.3	Classes de décalages	20
1.3.1	Décalages de type fini	20
1.3.2	Décalages sofiques	21
1.3.3	Décalages effectifs	23

Dans ce premier chapitre nous présentons les décalages dans le cas qui correspond au cadre le plus général de cette thèse, celui des des décalages définis sur un monoïde finiment présenté. La Partie 1.1 définit la notion de monoïde finiment présenté. Sur ces monoïdes il est possible de définir les décalages de deux façons équivalentes, qui correspondent à une approche topologique (Partie 1.2.1) ou combinatoire (Partie 1.2.2). La Partie 1.3 présente les trois classes de décalages dont il sera question dans la suite du manuscrit.

Ce choix de définir les décalages dans ce cadre très général permet de proposer un formalisme qui unifie les notations pour les décalages sur \mathbb{Z}^d (Chapitres 2 et 3) et des décalages d'arbre (Chapitre 4).

Dans ce chapitre nous utiliserons des notions élémentaires de théorie des groupes, et nous supposerons que le lecteur est familier de ces objets ; à défaut,

nous l'invitons à consulter [Rot94]. Nous supposerons de même pour la théorie des graphes pour laquelle nous conseillons [Die05].

1.1 Monoïdes finiment présentés

Cette partie est consacrée aux monoïdes finiment présentés, objets sur lesquels seront définis les décalages dans toute la suite. Notre étude se restreint à ces structures car elles constituent à nos yeux le juste milieu entre simplicité de la définition, puisque la description d'un tel monoïde est finie, et complexité de l'objet, puisque le problème du mot peut y être indécidable [Col86]. Un outil important pour l'étude de ces monoïdes est le graphe de Cayley (Partie 1.1.2), qui permet d'une part de visualiser leurs structures et d'autre part d'en caractériser certaines propriétés.

1.1.1 Définition

Un *monoïde* (\mathbb{M}, \circ) , ou plus simplement \mathbb{M} , est constitué d'un ensemble d'éléments \mathbb{M} stable par une loi de composition associative \circ , et qui possède un élément neutre $\varepsilon_{\mathbb{M}}$ pour cette loi. Nous prenons ici la convention de noter cette loi multiplicativement, en omettant la plupart du temps le symbole \circ .

Remarque. Dans toute la suite du manuscrit, nous ne considérons que des monoïdes dont le cardinal est infini.

Un monoïde \mathbb{M} est *libre* s'il existe un sous-ensemble \mathcal{G} de \mathbb{M} tel que tout élément de \mathbb{M} s'écrit d'une et une seule manière comme un produit fini d'éléments de \mathcal{G} . On dit alors que \mathbb{M} est *libre sur* \mathcal{G} . Tous les monoïdes libres sur un même ensemble \mathcal{G} sont isomorphes, c'est pourquoi on s'autorise à parler *du monoïde libre sur* \mathcal{G} . Dans le cas où \mathcal{G} est un ensemble fini à n éléments, on parle *du monoïde libre à n générateurs*, et on le note \mathbb{M}_n .

Soit \mathbb{M} le monoïde libre sur \mathcal{G} , et \mathcal{R} un sous-ensemble de $\mathbb{M} \times \mathbb{M}$, appelé ensemble de relations. On définit sur \mathbb{M} une relation symétrique $\mathcal{E} \subseteq \mathbb{M} \times \mathbb{M}$ de la façon suivante : on dit que $(g, h) \in \mathcal{E}$ si et seulement si $g = sut$ et $h = svt$ avec $v, s, t \in \mathbb{M}$ et $(u, v) \in \mathcal{R} \cup \{(m', m), (m, m') \in \mathcal{R}\}$. La clôture réflexive et transitive de la relation \mathcal{E} est alors une relation d'équivalence $\sim_{\mathcal{R}}$. En quotientant le monoïde libre \mathbb{M} par cette relation d'équivalence, on obtient le *monoïde de présentation* $\langle \mathcal{G} | \mathcal{R} \rangle$. On dit qu'un monoïde est *finiment présenté* s'il possède une présentation $\langle \mathcal{G} | \mathcal{R} \rangle$ pour laquelle \mathcal{G} et \mathcal{R} sont des ensembles finis. Le cardinal de l'ensemble \mathcal{G} est appelé le *rang du monoïde* \mathbb{M} .

Exemple 1.1.1. Le monoïde des mots finis sur l'alphabet $\{0, 1\}$ est un monoïde finiment présenté de rang 2 dont une présentation est $\langle 0, 1 \mid \emptyset \rangle$. L'opération dans ce monoïde est la concaténation, et l'élément neutre pour cette loi est le mot vide ε . Il est isomorphe au monoïde libre à deux générateurs \mathbb{M}_2 .

Exemple 1.1.2. Une présentation du groupe diédral d'ordre 4, qui est le groupe des isométries du carré, est :

$$\langle a, b \mid a^4 = b^2 = \varepsilon, ba = a^3b \rangle$$

où l'élément b peut être interprété comme une réflexion, et l'élément a une rotation d'angle $\frac{\pi}{4}$.

1.1.2 Graphe de Cayley

Le graphe de Cayley d'un groupe donne une représentation visuelle de la structure du groupe par un graphe. C'est aussi un moyen pour construire des graphes avec de fortes propriétés de régularité. On peut également étendre cette notion pour définir le graphe de Cayley d'un monoïde.

Définition 1. Le *graphe de Cayley du monoïde* $\mathbb{M} = \langle \mathcal{G} \mid \mathcal{R} \rangle$, noté $\Gamma_{\mathbb{M}} = (V_{\Gamma}, E_{\Gamma})$ est un graphe orienté construit de la manière suivante : l'ensemble des sommets est $V_{\Gamma} = \mathbb{M}$, et les arêtes sont

$$E_{\Gamma} = \{(g, g.g_i) \in \mathbb{M} \times \mathbb{M} \mid g_i \text{ est un générateur de } \mathbb{M}\}.$$

Notons que le graphe de Cayley du monoïde \mathbb{M} dépend donc de la représentation qu'on a choisie.

Deux exemples en sont donnés sur la Figure 1 ; pour plus de clarté on a attribué une couleur à chaque générateur du monoïde. Dans la suite nous identifierons un élément du monoïde \mathbb{M} avec le nœud qui le représente dans le graphe de Cayley $\Gamma_{\mathbb{M}}$. Lorsqu'aucune confusion n'est possible, le graphe de Cayley de \mathbb{M} sera simplement appelé Γ .

La *longueur* d'un élément g du monoïde \mathbb{M} est la longueur du plus court chemin entre les sommets correspondant à g et à l'élément neutre ε dans le graphe de Cayley Γ ; on la note $|g|$. Certaines propriétés du monoïde peuvent être lues directement sur le graphe de Cayley. De manière informelle, les cycles dans le graphe correspondent aux relations \mathcal{R} entre générateurs. En particulier, un graphe de Cayley est celui d'un monoïde libre si et seulement si le graphe est acyclique.

$$\mathbb{M}_2 = \langle 0, 1 / \emptyset \rangle$$

$$\mathcal{D}_4 = \langle a, b / a^4 = b^2 = \varepsilon, b.a = a^3.b \rangle$$

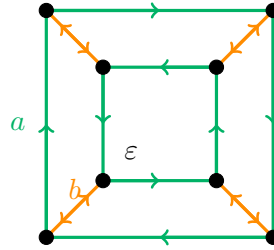
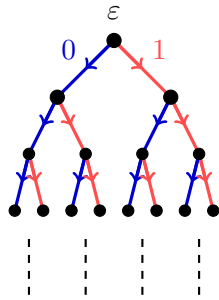


FIGURE 1: Les graphes de Cayley des monoïdes finiment présentés des Exemples 1.1.1 et 1.1.2.

1.1.3 Cas des groupes finiment présentés

Un groupe est un ensemble d'éléments muni d'une loi de composition interne associative, qui possède un élément neutre pour cette loi et tel que chaque élément possède un inverse. Cette structure gagne donc en rigidité par rapport à celle d'un monoïde car on impose la présence des inverses.

Remarque. Si \mathcal{G} est une partie d'un groupe, on notera par \mathcal{G}^{-1} l'ensemble des inverses des éléments de \mathcal{G} .

Nous avons choisi dans ce chapitre de d'abord traiter le cas des monoïdes finiment présentés. Toutes les notions définies précédemment sont toutefois compatibles avec les groupes finiment présentés, au sens où l'on peut voir un groupe finiment présenté $\mathbb{G} = \langle \mathcal{G} | \mathcal{R} \rangle$ comme un cas particulier de monoïde : si \mathcal{G} est un ensemble de générateurs pour le groupe, c'est-à-dire que tout élément du groupe s'écrit comme un produit fini d'éléments de \mathcal{G} et d'inverses d'éléments de \mathcal{G} , et que le groupe est défini par l'ensemble de relations \mathcal{R} sur des éléments de $\mathcal{G} \cup \mathcal{G}^{-1}$, alors le groupe \mathbb{G} peut être vu comme le monoïde finiment présenté suivant :

$$\mathbb{G} = \langle \mathcal{G} \cup \mathcal{G}^{-1} | \mathcal{R} \cup \{gg^{-1} = \varepsilon, g \in \mathcal{G}\} \rangle .$$

Le graphe de Cayley du groupe nous donne également un moyen très simple de définir une distance entre les éléments du groupe. Étant donnés deux éléments g et h du groupe \mathbb{G} , la *distance entre g et h* , notée $d_{\mathbb{G}}(g, h)$, est la longueur du plus court chemin entre les sommets correspondant à g et h dans le graphe de

Cayley $\Gamma(\mathbb{G})$ de \mathbb{G} . Cette distance est équivalente à la suivante :

$$d'(g, h) = \min \{ |j_1| + \dots + |j_k|, gh^{-1} = g_{i_1}^{j_1} \dots g_{i_k}^{j_k} \}.$$

De même que pour un monoïde, la *longueur* d'un élément g du groupe \mathbb{G} est la distance entre g et l'élément neutre ε ; on la note $|g| = d_{\mathbb{G}}(g, \varepsilon)$. Un résultat classique est que le groupe \mathbb{G} muni de la distance $d_{\mathbb{G}}$ est un groupe topologique. Notons qu'une fois encore la distance ainsi définie sur le groupe dépend de la présentation choisie, mais pour un groupe finiment présenté toutes les présentations donnent des distances équivalentes.

On notera \mathbb{F}_n le *groupe libre à n générateurs*. Par convention, lorsque l'on travaillera sur un groupe on omettra les inverses dans l'ensemble de générateurs.

1.2 Les décalages, deux points de vue complémentaires

Cette partie présente deux manières équivalentes de définir un décalage sur un monoïde finiment présenté. Ces deux définitions correspondent chacune à un certain point de vue, et sont donc plus ou moins adaptées selon le contexte.

1.2.1 Point de vue topologique

Dans cette partie, un décalage est vu comme un système dynamique, où l'on fait agir le monoïde \mathbb{M} sur un certain sous-ensemble de l'espace des configurations.

L'espace des configurations

L'espace des configurations $A^{\mathbb{M}}$ est défini comme un espace produit et à ce titre on peut le munir d'une topologie produit. Avec la topologie discrète (équivalente à celle donnée par la distance d définie par $d(x, y) = 0$ si $x = y$ et $d(x, y) = 1$ si $x \neq y$) sur l'alphabet A , on munit l'espace des configurations $A^{\mathbb{M}}$ d'une topologie produit (aussi appelée topologie pro-discrète) équivalente à celle donnée par la distance suivante :

$$\forall x, y \in A^{\mathbb{M}}, \begin{cases} d(x, y) = 2^{-\min\{|g|, x_g \neq y_g\}} & \text{si } x \neq y \\ d(x, y) = 0 & \text{si } x = y \end{cases}$$

Deux configurations sont d'autant plus proches qu'elles coïncident sur un motif central plus grand, et le plus court élément du monoïde pour lequel elles diffèrent donne la distance entre ces configurations.

Un résultat classique est que l'ensemble $A^{\mathbb{M}}$ des configurations, muni de cette topologie produit, est compact (il s'agit d'une application directe du théorème de Tychonoff).

Définition topologique des décalages

On définit la transformation $\sigma_{\mathbb{M}}$ comme l'action naturelle de \mathbb{M} sur l'espace des configurations, où un élément du monoïde $g \in \mathbb{M}$ agit sur $A^{\mathbb{M}}$ par translation (voir la Figure 2) via l'application $(\sigma_{\mathbb{M}})^g : A^{\mathbb{M}} \rightarrow A^{\mathbb{M}}$ (que l'on notera simplement σ^g lorsqu'aucune confusion n'est possible sur le monoïde \mathbb{M}) définie par :

$$(\sigma_{\mathbb{M}})^g(x)_h = x_{g,h} \text{ pour tout } i \in A^{\mathbb{M}}.$$

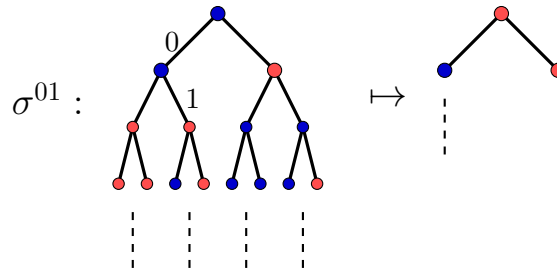


FIGURE 2: Action naturelle de $\mathbb{M}_2 = \{0, 1\}^*$ par translation sur l'espace des configurations $\{\bullet, \circ\}^{\mathbb{M}_2}$.

Si $x \in A^{\mathbb{M}}$ est une configuration et $\mathbb{S} \subseteq \mathbb{M}$ un ensemble de coordonnées, la restriction de x à \mathbb{S} notée $x_{\mathbb{S}}$ est un élément de $A^{\mathbb{S}}$ tel que :

$$\forall g \in \mathbb{S}, (x_{\mathbb{S}})_g = x_g.$$

Si \mathbb{S} est un sous-ensemble fini de $A^{\mathbb{M}}$ et $p \in A^{\mathbb{S}}$, on appelle *cylindre relatif* au motif p l'ensemble des configurations dans lesquelles le motif p apparaît en position ε :

$$[[p]] = \{x \in A^{\mathbb{M}} \mid x_{\mathbb{S}} = p\}.$$

Les cylindres sont des ouverts-fermés.

Définition 2. Un *décalage* est un sous-ensemble stable par la translation $\sigma_{\mathbb{M}}$ de $A^{\mathbb{M}}$, c'est-à-dire que $\sigma^g(\mathbf{T}) \subseteq \mathbf{T}$ pour tout $g \in \mathbb{M}$, et fermé pour la topologie produit.

On notera par $\mathcal{E}_A^{\mathbb{M}}$ l'ensemble de tous les décalages définis sur le monoïde \mathbb{M} et sur l'alphabet A . On notera aussi

$$\mathcal{E}^{\mathbb{M}} = \bigcup_{A \text{ alphabet fini}} \mathcal{E}_A^{\mathbb{M}}$$

l'ensemble des décalages sur le monoïde \mathbb{M} (on pourra omettre l'exposant \mathbb{M} lorsqu'aucune confusion n'est possible).

Remarque. Dans le cas d'un décalage \mathbf{T} sur un groupe \mathbb{G} , la stabilité par translation s'exprime de manière équivalente par $\sigma^g(\mathbf{T}) = \mathbf{T}$.

En particulier l'espace des configurations $A^{\mathbb{M}}$ est un décalage, que l'on appelle aussi *décalage plein*, ou *décalage trivial*.

1.2.2 Définition par motifs interdits

Dans cette partie nous optons pour le point de vue combinatoire pour définir les décalages. Toutes les définitions et propriétés exposées dans cette partie sont des généralisations de celles présentées dans l'ouvrage de référence de Lind et Marcus [LM95].

Dans toute cette partie nous considérons un monoïde finiment présenté $\mathbb{M} = \langle \mathcal{G} | \mathcal{R} \rangle$ de rang d , où $\mathcal{G} = \{g_1, \dots, g_d\}$ est un ensemble de d générateurs, ainsi qu'un alphabet fini A dont chaque élément est appelé *lettre* ou encore *couleur*.

Motifs et langages

On considère les coloriage de \mathbb{M} par A et on appelle *configuration* un élément de $A^{\mathbb{M}}$. Une configuration $x \in A^{\mathbb{M}}$ est donc une application $x : \mathbb{M} \rightarrow A$; on adopte dans la suite la notation x_g pour désigner l'image $x(g)$ de $g \in \mathbb{M}$ par l'application x . Un *support* est un ensemble fini $\mathbb{S} \subseteq \mathbb{M}$ qui contient le mot vide du monoïde ε . Étant donné un support \mathbb{S} , un *motif de support* \mathbb{S} est un élément de $A^{\mathbb{S}}$. De manière analogue, on désigne par p_g l'image de $g \in \mathbb{S}$ par l'application $p : \mathbb{S} \rightarrow A$.

On dit qu'un motif $p \in A^{\mathbb{S}}$ apparaît dans une configuration $x \in A^{\mathbb{M}}$ s'il existe un élément $g \in \mathbb{M}$ tel que pour tout $i \in \mathbb{S}$, $x_{g.i} = p_i$; dans ce cas on dit que le motif p apparaît dans x en position g , et on le note $p \sqsubset_g x$, ou simplement $p \sqsubset x$. On dit qu'un motif p de support \mathbb{S} apparaît dans un motif p' de support \mathbb{S}' s'il existe un élément $g \in \mathbb{S}'$ tel que pour tout $i \in \mathbb{S}$, $g.i \in \mathbb{S}'$ et $p'_{g.i} = p_i$; on note dans ce cas $p \sqsubseteq p'$. Si de plus pour tout $i \in \mathbb{S}$ et pour tout générateur $g_k \in \mathcal{G}$, $g.i.g_k \in \mathbb{S}'$, on dit que p apparaît strictement dans p' et on le note $p \sqsubset p'$. Le *support élémentaire de taille n* , noté \mathbb{S}_n , est l'ensemble des éléments

de \mathbb{M} situés à distance au plus n de l'élément neutre ε . Un *bloc de taille n* (ou *motif élémentaire de taille n*) est un motif dont le support est \mathbb{S}_n .

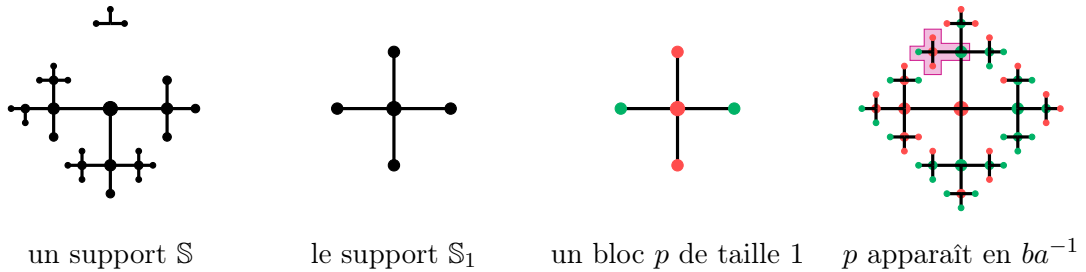


FIGURE 3: Sur le groupe libre à deux générateurs $\mathbb{F}_2 = \langle a, b | \emptyset \rangle$: un support, le support élémentaire de rayon 1, un bloc de taille 1 qui apparaît en position ba^{-1} .

Un *langage sur \mathbb{M}* est un ensemble de motifs dont le support est inclus dans \mathbb{M} , deux motifs du langage pouvant avoir des supports différents. Sans perte de généralité pour l'utilisation que nous allons en faire, comme nous le verrons dans la Partie 1.2.2 au moment de définir un décalage, on peut se ramener à ne considérer que des langages de motifs élémentaires.

Exemple 1.2.1. • Sur le monoïde \mathbb{N} et sur l'alphabet à deux éléments $\{\blacksquare, \blacksquare\}$, on définit le langage \mathcal{L}_1 comme l'ensemble fini de motifs élémentaires suivants :

$$\mathcal{L}_1 = \left\{ \begin{array}{c} \blacksquare \blacksquare \blacksquare \\ \blacksquare \blacksquare \blacksquare \end{array}, \begin{array}{c} \blacksquare \blacksquare \blacksquare \\ \blacksquare \blacksquare \blacksquare \end{array}, \begin{array}{c} \blacksquare \blacksquare \blacksquare \\ \blacksquare \blacksquare \blacksquare \end{array}, \begin{array}{c} \blacksquare \blacksquare \blacksquare \\ \blacksquare \blacksquare \blacksquare \end{array} \right\}$$

- Sur le groupe \mathbb{Z}^2 et sur l'alphabet $\{\blacksquare, \blacksquare\}$, on définit le langage \mathcal{L}_2 formé de tous les motifs élémentaires (dont le support est de la forme $[-n; n]^2$) présentant un \blacksquare au centre, entouré de \blacksquare , eux-mêmes entourés de \blacksquare , ...et ainsi de suite.

Motifs interdits et langage d'un décalage sur \mathbb{M}

Définition 3. Soit F un ensemble de motifs. On appelle *décalage défini par l'ensemble de motifs interdits F* l'ensemble des configurations

$$\mathbf{T}_F = \left\{ x \in A^{\mathbb{M}}, \text{ aucun motif de } F \text{ n'apparaît dans } x \right\}.$$

$$\mathcal{L}_2 = \left\{ \begin{array}{c} \text{■} , \begin{array}{|c|c|c|} \hline \text{■} & \text{■} & \text{■} \\ \hline \end{array} , \begin{array}{|c|c|c|c|} \hline \text{■} & \text{■} & \text{■} & \text{■} \\ \hline \end{array} , \begin{array}{|c|c|c|c|c|} \hline \text{■} & \text{■} & \text{■} & \text{■} & \text{■} \\ \hline \end{array} , \dots \end{array} \right\}$$

Notons que si F est un ensemble de motifs qui ne sont pas nécessairement élémentaires, on peut construire un ensemble \tilde{F} de motifs élémentaires tel que $\mathbf{T}_F = \mathbf{T}_{\tilde{F}}$. Pour cela il suffit de remplacer chaque motif $p \in A^{\mathbb{S}}$ de F par l'ensemble de motifs suivant :

$$\tilde{F} = \{ \tilde{p} \in A^{\mathbb{S}_n}, \tilde{p}_{\mathbb{S}} = p \}$$

où n est le plus petit entier tel que $\mathbb{S} \subseteq \mathbb{S}_n$.

Exemple 1.2.2. Le décalage \mathbf{T}_1 défini par le langage \mathcal{L}_1 de motifs interdits de l'Exemple 1.2.1 est l'ensemble des configurations de $\{ \text{■} , \text{■} \}^{\mathbb{N}}$ qui commencent par une suite (éventuellement infinie) de ■ et se terminent par des ■ :

$$\mathbf{T}_1 = \{ x \in \{ \text{■} , \text{■} \}^{\mathbb{N}}, \exists i \in \mathbb{N} \cup \{+\infty\}, (x_j = \text{■} \Leftrightarrow j \leq i) \}.$$

Ainsi à chaque ensemble de motifs F on peut associer un décalage, mais l'ensemble de motifs interdits qui définit un décalage n'est pas unique. Le décalage \mathbf{T}_1 de l'Exemple 1.2.2 peut aussi être défini par le seul motif interdit $F' = \{ \text{■} \text{■} \}$.

Définition 4. Le langage d'ordre n d'un décalage \mathbf{T} est l'ensemble des motifs élémentaires de taille n qui apparaissent dans les configurations de ce décalage.

$$\mathcal{L}_n(\mathbf{T}) = \{ p \in A^{\mathbb{S}_n}, p \text{ apparaît dans une configuration de } \mathbf{T} \}$$

Le langage d'un décalage \mathbf{T} est l'ensemble des motifs élémentaires qui apparaissent dans les configurations de ce décalage.

$$\mathcal{L}(\mathbf{T}) = \bigcup_{n \in \mathbb{N}} \mathcal{L}_n(\mathbf{T})$$

Tous les langages de motifs ne sont pas des langages de décalage. En effet, dès qu'un motif appartient au langage d'un décalage, alors ce motif apparaît dans une configuration. En conséquence tous ses sous-motifs, puisqu'ils apparaissent dans la même configuration, sont aussi dans le langage du décalage. De même, si un motif appartient au langage il apparaît dans une configuration infinie, donc *a fortiori* il apparaît strictement dans un autre motif du langage. Ces deux notions sont en fait suffisantes pour caractériser les langages de décalage, comme nous le montrons dans la proposition suivante.

Proposition 1.2.1. *Soit \mathcal{L} un langage de motifs sur un monoïde \mathbb{M} . Alors \mathcal{L} est le langage d'un décalage si et seulement si*

- \mathcal{L} est factoriel : $\forall m \in \mathcal{L}$, si $m' \sqsubseteq m$ alors $m' \in \mathcal{L}$;
- \mathcal{L} est prolongeable : $\forall m \in \mathcal{L}$, $\exists m' \in \mathcal{L}$, $m \sqsubset m'$.

Le langage \mathcal{L}_1 de l'Exemple 1.2.1 n'est ni factoriel ni prolongeable (comme tout autre langage fini pour ce dernier point), tandis que le langage \mathcal{L}_2 du même exemple est prolongeable mais pas factoriel car il ne contient pas le motif \blacksquare par exemple.

Démonstration. • Soit $\mathcal{L} = \mathcal{L}(\mathbf{T})$ le langage d'un décalage. Si $m \in \mathcal{L}$ et $m' \sqsubseteq m$, alors il existe $x \in \mathbf{T}$ tel que $m \sqsubseteq x$, et donc *a fortiori* $m' \sqsubseteq x$. Ainsi $m' \in \mathcal{L}$, c'est-à-dire que \mathcal{L} est stable par sous-mot élémentaire. D'autre part si $m \in \mathcal{L}$ tel que $m \in \mathcal{L}_n(\mathbf{T})$, il existe $x \in \mathbf{T}$ tel que $x|_{\mathbb{S}_n} = m$. On pose $m' = x|_{\mathbb{S}_{n+1}}$. Alors $m \sqsubset m'$ et $m' \in \mathcal{L}$ donc \mathcal{L} est prolongeable.

- Réciproquement soit $\mathcal{L} \subset \mathcal{E}$ un langage prolongeable et stable par sous-mot élémentaire. Considérons le décalage $\mathbf{T} = \mathbf{T}_{\mathcal{L}^c}$ et montrons que $\mathcal{L} = \mathcal{L}(\mathbf{T})$.
 - Si $m \in \mathcal{L}(\mathbf{T})$, alors $m \sqsubseteq x \in \mathbf{T}_{\mathcal{L}^c}$ donc $m \notin \mathcal{L}^c$ i.e. $m \in \mathcal{L}$. On a bien $\mathcal{L}(\mathbf{T}) \subseteq \mathcal{L}$.
 - Si $m \in \mathcal{L}$, alors il existe un entier n tel que $m \in \mathcal{A}^{\mathbb{S}_n}$. Comme le langage \mathcal{L} est prolongeable, il existe $m_1 \in \mathcal{A}^{\mathbb{S}_{n+1}} \cap \mathcal{L}$ tel que $m \sqsubset m_1$. En itérant le procédé, on construit une suite $(m_k)_{k \in \mathbb{N}}$ d'éléments de \mathcal{L} telle que $m \sqsubset m_1 \sqsubset \dots \sqsubset m_k \sqsubset m_{k+1} \sqsubset \dots$. Notons x_k un élément de $\mathcal{A}^{\mathbb{M}}$ tel que $(x_k)|_{\mathbb{S}_{n+k}} = m_k$. Par compacité de l'espace des configurations $\mathcal{A}^{\mathbb{M}}$, il existe ϕ une extraction telle que la suite extraite $(x_{\phi(k)})_{k \in \mathbb{N}}$ converge ; on appelle $x = \lim x_{\phi(k)}$, montrons que $x \in \mathbf{T}$. Soit m' un motif qui apparait dans x . Alors $\exists k \in \mathbb{N}$ tel que

$m' \sqsubseteq m_k$, donc $m' \notin \mathcal{L}^c$. On a bien $x \in \mathbf{T}$ donc m est autorisé dans \mathbf{T} c'est-à-dire $\mathcal{L} \subseteq \mathcal{L}(\mathbf{T})$. □

Toute naturelle que semble être la notion de langage d'un décalage, elle n'en est pas moins extrêmement contraignante comme nous venons de le voir. C'est d'ailleurs un raison pour laquelle on définit habituellement un décalage en donnant un ensemble de motifs interdits et non son langage.

Le langage $\mathcal{L}(\mathbf{T})$ d'un décalage \mathbf{T} est donc exactement l'ensemble de tous les motifs autorisés (par opposition aux motifs interdits). Son complémentaire, que l'on notera $\mathcal{L}(\mathbf{T})^c$, est donc exactement l'ensemble de tous les motifs interdits c'est-à-dire le plus grand ensemble (au sens de l'inclusion) de motifs interdits qui définit \mathbf{T} .

Proposition 1.2.2. *Soit \mathbf{T} un décalage. Alors le complémentaire du langage $\mathcal{L}(\mathbf{T})^c$ de \mathbf{T} est un ensemble de motifs interdits qui définit $\mathbf{T} : \mathbf{T} = \mathbf{T}_{\mathcal{L}(\mathbf{T})^c}$.*

Démonstration. Soit x une configuration du décalage \mathbf{T} . Comme le langage $\mathcal{L}(\mathbf{T})$ contient par définition tous les motifs élémentaires apparaissant dans \mathbf{T} , tous les motifs élémentaires qui apparaissent dans x sont dans $\mathcal{L}(\mathbf{T})$, c'est-à-dire que x est bien une configuration du décalage défini par l'ensemble de motifs interdits $\mathcal{L}(\mathbf{T})^c$. Ainsi x est bien une configuration du décalage $\mathbf{T}_{\mathcal{L}(\mathbf{T})^c}$.

Réciproquement supposons que x soit une configuration du décalage $\mathbf{T}_{\mathcal{L}(\mathbf{T})^c}$. Alors tout motif élémentaire qui apparaît dans x est dans $\mathcal{L}(\mathbf{T})$. En particulier pour tout $n \in \mathbb{N}$, le motif élémentaire $x_{\mathbb{S}_n}$ de taille n est dans le langage de \mathbf{T} . Par conséquent il existe une configuration x_n dans le décalage \mathbf{T} qui coïncide avec x sur le support $\mathbb{S}_n : (x_n)_{|\mathbb{S}_n} = x_{|\mathbb{S}_n}$. On peut construire une configuration $x_n \in \mathbf{T}$ pour tout entier n , de sorte que la suite des x_n converge vers la configuration x . Le décalage \mathbf{T} étant fermé, on en déduit que x est bien une configuration de \mathbf{T} . □

1.2.3 Coïncidence des définitions

Proposition 1.2.3. *Les définitions combinatoire et topologique coïncident.*

Démonstration. Supposons que \mathbf{T} est un décalage au sens combinatoire de la Définition 3. Alors il existe un ensemble de motifs interdits F tel que $\mathbf{T} = \mathbf{T}_F$. En réécrivant le décalage \mathbf{T} à l'aide de cylindres on a :

$$\mathbf{T}_F = \bigcap_{m \in F} \bigcap_{u \in \mathbb{M}} \sigma^u(\llbracket m \rrbracket^c) = \bigcap_{m \in F} \bigcap_{g \in \mathbb{M}} (\sigma_{\mathbb{M}}^g(\llbracket m \rrbracket))^c = A^{\mathbb{M}} \setminus \bigcup_{m \in F} \bigcup_{g \in \mathbb{M}} \sigma_{\mathbb{M}}^g(\llbracket m \rrbracket)$$

Cette écriture permet de retrouver facilement la définition topologique : la stabilité par translation $\sigma_{\mathbb{M}}$ apparaît clairement, et le décalage \mathbf{T} est un ensemble fermé comme intersection de fermés.

Réciproquement supposons que \mathbf{T} soit un décalage sur \mathbb{M} au sens topologique de la Définition 2. Alors par la Proposition 1.2.2 on sait que le complémentaire du langage de \mathbf{T} est un ensemble de motifs interdits qui le définit. \square

La première définition, topologique, (voir la Partie 1.2.1) provient directement de la théorie des systèmes dynamiques. Un décalage est vu comme un ensemble fermé et stable par l'action naturelle du monoïde (par translation). Cette définition donne aussi un critère simple pour vérifier qu'un ensemble de configurations est bien un décalage, sans toutefois avoir besoin d'exhiber un ensemble de motifs interdits.

La deuxième définition, combinatoire, (voir la Partie 1.2.2) est utilisable très concrètement. Elle permet notamment de construire un décalage en explicitant un ensemble de motifs interdits ; chaque motif interdit ou autorisé correspond en fait à une règle locale que les configurations du décalage doivent respecter. La plupart des exemples présentés dans ce manuscrit utilisent cette définition. Comme nous le verrons par la suite, c'est aussi la définition qui se rapproche le plus de celles des pavages du plan discret \mathbb{Z}^2 et des diagrammes espace-temps de machines de Turing, objets sur lesquels nous reviendrons plus en détails dans la Partie 2.3.1.

1.3 Classes de décalages


Cette partie introduit les trois classes de décalages dont il sera question dans la suite de cette thèse. Les deux premières, la classe des décalages de type fini et la classe des décalages sofiques, sont l'expression de contraintes locales sur les configurations. La troisième, celle des décalages effectifs, apparaîtra naturellement dans la Partie 2.2.3 en appliquant l'opération de sous-action projective aux deux classes précédentes.

1.3.1 Décalages de type fini

Les décalages triviaux sont les ensembles de configurations $A^{\mathbb{M}}$, on note par \mathcal{FS} (de l'anglais *full-shift*) la classe des décalages triviaux. Les décalages de type fini, dont les configurations vérifient un nombre fini de contraintes locales, sont les décalages les plus simples à définir et les plus aisément manipulables (décalages triviaux mis à part).

Définition 5. Un décalage est dit *de type fini* si on peut le définir par un ensemble fini de motifs interdits. On note par \mathcal{SFT} (de l'anglais *shift of finite type*) la classe des décalages de type fini.

Si \mathbf{T} est un décalage de type fini donné par l'ensemble F de motifs interdits, on peut supposer que tous les motifs de F ont le même support. Si ce n'est pas le cas, on transforme F en un ensemble de motifs interdits de même support de la manière suivante. Soit k la taille du plus grand support d'un motif de F . Pour tout motif $m \in F$, on remplace m par l'ensemble de motifs $\mathcal{L}_k([m])$ de support \mathbb{S}_k . Si F est un ensemble de motifs qui sont tous de support \mathbb{S}_k , on dit que l'ensemble F est d'ordre k . Étant donné un décalage de type fini \mathbf{T} , le plus petit entier pour lequel on peut trouver un ensemble d'ordre k qui définit \mathbf{T} est appelé *l'ordre* du décalage \mathbf{T} .

Exemple 1.3.1. Le décalage \mathbf{T}_1 de l'Exemple 1.2.2 est un décalage de type fini d'ordre 2, puisqu'il suffit d'interdire le motif  pour le définir.

1.3.2 Décalages sofiques

On considère un nouvel alphabet A' , pas nécessairement différent de A , ainsi que \mathbf{T} un décalage sur l'alphabet A et m un entier naturel. On définit tout d'abord les fonctions de bloc, qui permettent de recoder l'alphabet A en un nouvel alphabet A' , ce recodage se faisant en respectant des règles locales.

Définition 6. Une application $\Phi : \mathbf{T} \subseteq A^{\mathbb{M}} \rightarrow A'^{\mathbb{M}}$ est appelée *m-fonction de bloc* s'il existe une application locale $\phi : \mathcal{L}_m(\mathbf{T}) \rightarrow A'$ telle que

$$\text{pour tout } g \in \mathbb{M}, \Phi(x)_g = \phi(x_{|_{g.\mathbb{S}_m}}),$$

où $x_{|_{g.\mathbb{S}_m}}$ est le motif de support \mathbb{S}_m qui apparaît en position g dans la configuration x (voir la Figure 4). On dit dans ce cas que ϕ est la *fonction locale* de Φ . Le plus petit entier m vérifiant cette propriété est appelé *la mémoire* de la fonction de bloc Φ .

Dans le cas où $\mathbb{M} = \mathbb{Z}$, le théorème de Curtis-Lyndon-Hedlund (voir Théorème 3.4 de [Hed69]) nous apprend que les fonctions de blocs sont exactement les applications $\Phi : X \rightarrow Y$ qui sont continues et qui commutent avec l'action $\sigma_{\mathbb{Z}}$. Cette propriété reste vraie dans le cas d'un monoïde finiment présenté.

Proposition 1.3.1. *Les fonctions de bloc sur \mathbb{M} sont exactement les applications $\Phi : A^{\mathbb{M}} \rightarrow A'^{\mathbb{M}}$ qui sont continues et qui commutent avec la translation $\sigma_{\mathbb{M}}$.*

Cette caractérisation des fonctions de bloc nous permet de montrer facilement qu'elles transforment des décalages en décalages.

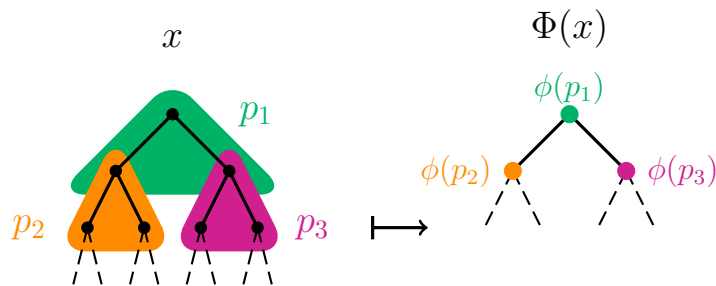


FIGURE 4: Une 1-fonction de bloc Φ appliquée à une configuration sur \mathbb{M}_2 . L'image de la configuration x par Φ est calculée grâce à sa fonction locale ϕ . Dans $\Phi(x)$, la lettre d'un nœud dépend de la lettre de ce nœud et de celles de ses deux fils dans la configuration x .

Proposition 1.3.2. *L'image d'un décalage par une fonction de bloc est également un décalage.*

Démonstration. Considérons un décalage \mathbf{T} et une fonction de bloc Φ de mémoire m . Il suffit de montrer que l'ensemble $\Phi(\mathbf{T})$ est fermé et invariant par σ^g pour tout élément $g \in \mathbb{M}$. Comme Φ commute avec σ^g , $\sigma^g(\Phi(\mathbf{T})) = \Phi(\sigma^g(\mathbf{T}))$. Le décalage \mathbf{T} commute avec σ^g , on a donc $\Phi(\sigma^g(\mathbf{T})) = \Phi(\mathbf{T})$. L'ensemble $\Phi(\mathbf{T})$ est donc invariant par σ^g . Enfin, $\Phi(\mathbf{T})$ étant l'image continue d'un ensemble compact, il s'agit d'un ensemble fermé et donc d'un décalage. \square

Proposition 1.3.3. *L'inverse d'une fonction de bloc bijective entre deux décalages est également une fonction de bloc.*

Démonstration. Soit Φ une fonction de bloc bijective. Alors Φ possède un inverse Φ^{-1} , et il est facile de voir que cette fonction inverse commute avec la translation $\sigma_{\mathbb{M}}$. Par compacité de l'espace des configurations $A^{\mathbb{M}}$, la fonction Φ^{-1} est continue. D'après la Proposition 1.3.1, la fonction Φ^{-1} est donc aussi une fonction de bloc. \square

Une fonction de bloc bijective entre \mathbf{T} et \mathbf{T}' est appelée une *conjugaison*. On dit dans ce cas que les décalages \mathbf{T} et \mathbf{T}' sont *conjugués*. Si la k -fonction de bloc Φ est une conjugaison, on dit que c'est une *k -conjugaison*.

Le résultat de la Proposition 1.3.2 nous permet de définir une nouvelle classe de décalages, en nous intéressant à l'image des SFT par les fonctions de bloc. Les SFT sont des objets que l'on peut décrire de manière très simple, en énumérant un ensemble fini de motifs interdits. De la même façon, une fonction de bloc

se décrit en énumérant les règles de la fonction locale qui lui correspond. Les décalages sofiqes possèdent donc eux aussi une description finie.

Définition 7. Un décalage est dit *sofique* s'il est l'image par une fonction de bloc d'un décalage de type fini. On note par *Sofique* la classe des décalages sofiqes.

La classe des décalage sofique est stable par fonction de bloc : en composant deux fonctions de bloc on obtient encore une fonction de bloc. Cette nouvelle classe contient strictement la classe des décalages de type fini, comme le montre l'Exemple 1.3.2.

Exemple 1.3.2. On se place sur l'alphabet $A = \{\square, \blacksquare, \blacktriangle\}$. On définit le décalage de type fini $\mathbf{T} = \mathbf{T}_{\{\blacksquare\blacksquare, \blacktriangle\blacksquare, \blacksquare\square, \blacktriangle\square\}} \subset \mathcal{A}^{\mathbb{Z}}$. Soit $\Pi : A^{\mathbb{Z}} \rightarrow B^{\mathbb{Z}}$ la fonction de bloc définie localement par $\pi : \begin{cases} \square \mapsto \square \\ \blacksquare \mapsto \blacktriangle \\ \blacktriangle \mapsto \blacktriangle \end{cases}$. La configuration suivante est bien dans le décalage \mathbf{T} :

$$x = \dots \square\square\blacksquare\blacksquare\blacktriangle\square\square\square\blacksquare\square\square\blacksquare\blacksquare\blacktriangle\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\square\square\square\blacksquare\blacksquare\dots$$

et si on lui applique la fonction de bloc Π , on obtient la configuration

$$\Pi(x) = \dots \square\square\blacktriangle\blacktriangle\blacktriangle\blacktriangle\square\square\square\blacktriangle\blacktriangle\blacktriangle\blacktriangle\blacktriangle\blacktriangle\blacktriangle\blacktriangle\blacktriangle\square\square\square\blacktriangle\blacktriangle\dots$$

Ainsi le décalage sofique obtenu en appliquant la fonction de bloc Π sur le décalage de type fini \mathbf{T} est :

$$\Pi(\mathbf{T}) = \{x \in \{\square, \blacktriangle\}^{\mathbb{Z}} \mid \text{les blocs de } \blacktriangle \text{ sont de longueur paire}\}$$

que l'on peut décrire par l'ensemble de motifs interdits suivant :

$$\Pi(\mathbf{T}) = \mathbf{T}_{\{\square\blacktriangle^{2n+1}\square : n \in \mathbb{N}\}}.$$

Ce décalage $\Pi(\mathbf{T})$ n'est pas de type fini, c'est l'exemple le plus classique connu sous le nom de *décalage pair* (en anglais *even subshift*).

Dans le Chapitre 4 nous nous intéresserons notamment aux décalages sofiqes sur le monoïde libre \mathbb{M}_2 .

1.3.3 Décalages effectifs

Une *machine de Turing* avec ruban \mathbb{M} est un modèle de calcul formé d'une tête de calcul (un automate fini) qui se déplace sur le graphe obtenu en retirant l'orientation des arêtes du graphe de Cayley du monoïde \mathbb{M} [GM07]. Étant donné un monoïde finiment présenté \mathbb{M} , on note $\mathcal{T}_{\mathbb{M}} = (V, E)$ ce graphe non orienté.

Définition 8. Une machine de Turing avec ruban \mathbb{M} est la donnée de $\mathcal{M} = (Q, \Gamma, \#, q_0, \delta, Q_F)$ où :

- Q est un l'ensemble fini des états possibles pour la tête de calcul ; $q_0 \in Q$ est l'état initial ;
- Γ est un alphabet fini ;
- $\# \notin \Gamma$ est le symbole blanc ;
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times (E \cup \{\varepsilon\})$ est la fonction de transition, où E est l'ensemble des arêtes du graphe non orienté $\mathcal{T}_{\mathbb{M}}$ associé au monoïde \mathbb{M} . Étant donné un état de la tête de calcul et la lettre inscrite sur le ruban à la position de la tête de calcul, cette lettre est remplacée par une nouvelle lettre, la tête de calcul se déplace sur une cellule adjacente (ou bien reste en place) et passe dans un nouvel état ;
- $F \subset Q_F$ est l'ensemble des états finaux. Lorsqu'un état final est atteint, la machine arrête son calcul.

Une *configuration* de la machine \mathcal{M} est la description de son ruban à une étape de calcul donnée. Il s'agit donc d'un élément de l'ensemble $(\Gamma \cup (Q \times \Gamma))^{\mathbb{M}}$, avec la contrainte de n'avoir qu'une seule tête de lecture, et seulement un nombre fini de cellules qui ne contiennent pas le symbole blanc $\#$. Un *calcul* de la machine \mathbb{M} sur le motif d'entrée p de support \mathbb{S} est la suite de configurations (c_n) , où c_0 est la configuration telle que pour tout $g \in \mathbb{M} \setminus \mathbb{S}$, $(c_0)_g = \#$ et pour tout $g \in \mathbb{S}$, $(c_0)_g = p_g$, et les configurations suivantes sont telles qu'on peut passer de la configuration c_n à la configuration c_{n+1} en appliquant la fonction de transition de la machine \mathcal{M} . Un calcul de \mathcal{M} sur un motif d'entrée p est fini si et seulement si un état final est atteint ; on dit dans ce cas que la machine \mathcal{M} *s'arrête* sur le motif d'entrée p . On appelle *domaine* de la machine \mathcal{M} l'ensemble des motifs d'entrée sur lesquels la machine \mathcal{M} s'arrête.

Lorsque le monoïde \mathbb{M} est \mathbb{N} (resp. \mathbb{Z}), on retrouve la définition classique des machines de Turing avec ruban semi-infini (resp. bi-infini) [Rog87]. La thèse de Church propose de formaliser la notion de calcul effectif par ces machine de Turing classiques. Cette hypothèse est largement confortée par le fait que les différents modèles de calcul proposés pour définir cette notion sont de puissance équivalente (programmes RAM, lambda-calcul, fonctions récursives). Cependant lorsque l'on définit des machines de Turing ayant comme ruban le graphe de Cayley d'un monoïde \mathbb{M} ou groupe \mathbb{G} finiment présenté quelconque, la puissance de calcul de ces machines peut être strictement supérieure à celle des machines de Turing classiques (une machine de Turing classique lit les lettres d'un motif sur \mathbb{M} ou \mathbb{G} dans l'ordre lexicographique de leur position).

Théorème 1.3.4 (Folklore). *Les machines de Turing sur \mathbb{G} ont même puissance de calcul que les machines de Turing classique si et seulement si le problème du mot est décidable sur \mathbb{G} .*

Dans les cas particuliers étudiés dans cette thèse (décalages sur \mathbb{Z}^d et décalages sur \mathbb{M}_2), les machines de Turing peuvent donc toujours être pensées comme des machines de Turing au sens classique.

Étant donné un langage de motifs \mathcal{L} , une machine de Turing \mathcal{M} peut être vue de deux façons comme un reconnaisseur de ce langage.

Définition 9. On dit que la machine \mathcal{M} *décide*, ou *reconnaît*, le langage \mathcal{L} si la machine \mathcal{M} s'arrête sur tout motif d'entrée, et répond de manière positive si le motif d'entrée appartient au langage \mathcal{L} , de manière négative sinon. On dit alors que le langage \mathcal{L} est *récuratif*.

Définition 10. On dit que la machine \mathcal{M} *énumère* le langage \mathcal{L} si la machine \mathcal{M} s'arrête exactement sur les motifs du langage \mathcal{L} . On dit dans ce cas que le langage \mathcal{L} est *récurativement énumérable*.

Remarque. Tout langage récuratif est aussi récurativement énumérable, mais la réciproque est fautive.

Définition 11. Un décalage \mathbf{T} est *effectif* s'il existe un ensemble de motifs récurativement énumérable F qui le définit, c'est-à-dire tel que $\mathbf{T} = \mathbf{T}_F$. On appelle \mathcal{RE} la classe des décalages effectifs.

On pourrait définir de manière analogue la classe des décalages récuratifs comme la classe des décalages qui peuvent être définis par un ensemble récuratif de motifs interdits. Cette classe est incluse dans la classe des décalages effectifs.

Proposition 1.3.5. *Pour tout décalage effectif \mathbf{T} , il existe un ensemble récuratif de motifs interdits F tel que $\mathbf{T} = \mathbf{T}_F$.*

Démonstration. Soit \mathbf{T} un décalage effectif défini sur un alphabet A . Alors il existe un ensemble récurativement énumérable F de motifs interdits tel que $\mathbf{T} = \mathbf{T}_F$. Sans perte de généralité on peut supposer que F est le complémentaire du langage de \mathbf{T} , c'est-à-dire $F = \mathcal{L}(\mathbf{T})^c$. Soit $(m_k)_{k \in \mathbb{N}}$ un énumération de l'ensemble F . On peut lui associer la suite $(n_k)_{k \in \mathbb{N}}$ des tailles des supports des motifs m_k . Si la suite $(n_k)_{k \in \mathbb{N}}$ est croissante, alors l'ensemble F est aussi récuratif (voir [Rog87, Théorème III p. 59]) et donc la proposition est démontrée. Sinon, on construit un nouvel ensemble \tilde{F} qui possède une énumération $(\tilde{m}_k)_{k \in \mathbb{N}}$ telle

que la suite des tailles de support $(\tilde{n}_k)_{k \in \mathbb{N}}$ est croissante, et qui définit le même décalage \mathbf{T} .

Pour tout motif m_k on s'intéresse à l'entier $\max_{i \leq k} n_i$. Si cet entier est égal à n_k , c'est-à-dire que tous les motifs qui précèdent m_k dans la suite sont définis sur des supports de taille inférieure ou égale à celle du support de m_k , alors le motif m_k apparaîtra tel quel dans la suite $(\tilde{m}_k)_{k \in \mathbb{N}}$, pas nécessairement au même rang. Si l'entier $\max_{i \leq k} n_i$ est différent de n_k , alors le motif m_k est précédé dans la suite par un motif dont le support est de taille strictement supérieure. Dans la nouvelle suite $(\tilde{m}_k)_{k \in \mathbb{N}}$, le motif m_k sera remplacé par une suite de motifs $\tilde{m}_{\phi(k)}, \dots, \tilde{m}_{\phi(k)+\psi(k)}$ de même support de taille $\max_{i \leq k} n_i$. Les fonctions ϕ et ψ dépendent de la suite $(n_k)_{k \in \mathbb{N}}$ et sont définies récursivement de la manière suivante :

- $\phi(0) = 0$ et $\psi(0) = 0$;
- pour tout entier n la fonction ϕ est donnée par $\phi(n+1) = \phi(n) + \psi(n) + 1$.
La fonction ψ est donnée par :

$$\psi(n+1) = \begin{cases} 0 & \text{si } n_k = \max_{i \leq k} n_i \\ (|\mathbb{S}_{\max_{i \leq k} n_i}| - |\mathbb{S}_{n_k}|)^A & \text{si } n_k \neq \max_{i \leq k} n_i. \end{cases}$$

Les motifs $\tilde{m}_{\phi(k)}, \dots, \tilde{m}_{\phi(k)+\psi(k)}$ sont l'ensemble des motifs de support $\mathbb{S}_{\max_{i \leq k} n_i}$ dans lesquels le motif m_k apparaît en position ε .

Ainsi le nouvel ensemble de motifs interdits \tilde{F} est récursif, et tel que $\tilde{F} \subseteq F$. Plus précisément, tout motif de F apparaît dans un motif de \tilde{F} . Il définit aussi le décalage \mathbf{T} . Soit x une configuration du décalage $\mathbf{T}_{\tilde{F}}$. Alors x ne peut pas contenir de motif de F , car sinon il contiendrait un motif de \tilde{F} ce qui est contradictoire. Donc $\mathbf{T} = \mathbf{T}_{\tilde{F}}$ ce qui montre la proposition. \square

La Proposition 1.3.5 montre qu'en fait la classe des décalages récursifs est exactement celle des décalages effectifs, c'est pourquoi dans la suite de cette thèse il ne sera plus question de décalages récursifs.

Exemple 1.3.3. On construit un exemple de décalage sur \mathbb{Z} effectif qui n'est pas sofique. On se place sur l'alphabet $A = \{0, a, b\}$, et on considère le décalage $\mathbf{T} = \mathbf{T}_{\{ba; 0a^m b^n 0; m \neq n\}}$. Alors son langage est

$$\mathcal{L}(\mathbf{T}) = \{m \in \{0, a, b\}^* \mid \text{les blocs contenant des } a \text{ et des } b \text{ sont de la forme } a^n b^n\},$$

qui est un exemple classique de langage non rationnel. Donc le décalage \mathbf{T} est un décalage effectif non sofique.

La classe des décalages effectifs correspond donc aux décalages calculables. Elle contient les deux classes définies précédemment, de manière stricte sur \mathbb{Z} comme le montrent les exemples 1.3.3 et 1.3.2 :

$$\mathcal{SFT} \subseteq \mathcal{Sofique} \subseteq \mathcal{RE}.$$

Dans la Partie 2.2.3 du Chapitre 2 consacré aux décalages sur \mathbb{Z}^d , l'étude de la sous-action projective fera apparaître les décalages effectifs comme ceux obtenus en appliquant cette opérations aux décalages sofiques.

2 Codage d'une machine de Turing dans un décalage sur \mathbb{Z}^2

2.1	Décalages substitutifs et décalages S-adiques	30
2.1.1	Substitutions	30
2.1.2	Composition de substitutions	33
2.1.3	Décalages S-adiques	33
2.1.4	Décalages substitutifs non déterministes	34
2.1.5	Substitutions non-déterministes et théorème de Mozes	36
2.2	Opérations et simulation	38
2.2.1	Simulation et classes stables	39
2.2.2	Exemples d'opérations et résultats classiques de simulation	39
2.2.3	La sous-action projective	41
2.3	Décalage sofique codant une machine de Turing . .	46
2.3.1	Machines de Turing dans un SFT	46
2.3.2	Espaces de calcul pour MT	50
2.3.3	Communication dans le décalage $\mathbf{T}_{\{s_1\}}$	53
2.3.4	Initialisation des calculs	55
2.3.5	Diagramme espace-temps d'une MT dans un décalage sofique	57

Ce chapitre explique comment construire, pour toute machine de Turing avec ruban semi-infini, un décalage sofique sur \mathbb{Z}^2 qui contient les diagrammes espace-temps de cette machine (Théorème 2.3.4).

Cette construction utilise un décalage substitutif (voir la Partie 2.1), dont on sait qu'il est aussi sofique par un résultat de Mozes [Moz89], pour définir des espaces de calcul. Les diagrammes espace-temps de la machine de Turing sont ajoutés à ces espaces de calcul à l'aide d'opérations simples sur les décalages (voir la Partie 2.2) afin d'obtenir le décalage sofique souhaité (voir la Partie 2.3).

Cette construction constitue le point principal de la preuve du Théorème 3.1.1, qui sera présentée dans le Chapitre 3. Elle sera aussi réutilisée et adaptée aux machines de Turing à semi-oracle dans la Partie 3.2.1.

2.1 Décalages substitutifs et décalages S-adiques

Cette partie est consacrée à la définition des décalages substitutifs, qui seront utilisés dans la construction de la Partie 2.3, ainsi que des décalages S-adiques dont il sera à nouveau question dans la Partie 3.2.2.

2.1.1 Substitutions

Soit $\mathbf{n} = (\mathbf{n}_1, \dots, \mathbf{n}_d) \in \mathbb{N}^d$ et $\mathbf{k} = (\mathbf{k}_1, \dots, \mathbf{k}_d) \in \mathbb{N}^d$, on définit $\mathbf{n} + \mathbf{k} = (\mathbf{n}_1 + \mathbf{k}_1, \dots, \mathbf{n}_d + \mathbf{k}_d) \in \mathbb{N}^d$. Étant donné $\mathbf{k} = (\mathbf{k}_1, \dots, \mathbf{k}_d)$, on appelle $\mathbb{U}_{\mathbf{k}}$ le rectangle $[0; \mathbf{k}_1] \times [0; \mathbf{k}_2] \times \dots \times [0; \mathbf{k}_d]$.

Sur un alphabet fini A , l'ensemble des *motifs rectangulaires* est $\mathcal{P} = \bigcup_{\mathbf{k} \in \mathbb{N}^d} A^{\mathbb{U}_{\mathbf{k}}}$. Une *substitution de dimension d* (ou *substitution multidimensionnelle*) est une fonction $s : A \rightarrow \mathcal{P}$. À toute lettre $a \in A$, on associe le vecteur $\mathbf{k}^s(a) = (\mathbf{k}_1^s(a), \dots, \mathbf{k}_d^s(a))$ tel que $\text{supp}(s(a)) = \mathbb{U}_{\mathbf{k}^s(a)}$, ce qui signifie qu'avec notre formalisme, le support du motif $s(a)$ dépend de la lettre a . Une substitution multidimensionnelle est *non dégénérée* si $\mathbf{k}_l^s(a) \geq 1$ pour tout $l \in [1; d]$ et pour toute lettre $a \in A$.

Soit $(\mathbf{k}^n)_{n \in \mathbb{Z}}$ une suite de vecteurs d -dimensionnels. Pour tout $n \in \mathbb{Z}$ on définit une fonction

$$\phi^{(\mathbf{k}^n)_{n \in \mathbb{Z}}} : \begin{pmatrix} \mathbb{Z}^d & \rightarrow & \mathbb{Z}^d \\ \mathbf{i} & \mapsto & (\phi_1(\mathbf{i}_1), \phi_2(\mathbf{i}_2), \dots, \phi_d(\mathbf{i}_d)) \end{pmatrix}$$

où $\phi_l(r) = \sum_{j=0}^{r-1} (\mathbf{k}^j)_l$ si $r \geq 0$ et $\phi_l(r) = \sum_{j=r}^{-1} (\mathbf{k}^j)_l$ si $r < 0$.

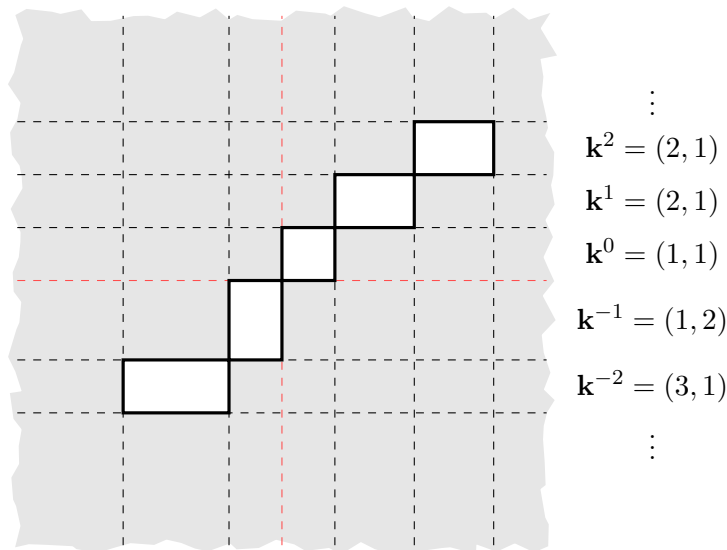


FIGURE 5: Un exemple de suite $(\mathbf{k}^n)_{n \in \mathbb{Z}}$ de vecteurs 2-dimensionnels qui déforme le réseau \mathbb{Z}^2 .

Cette fonction $\phi^{(\mathbf{k}^n)_{n \in \mathbb{Z}}}$ permet de déformer le réseau \mathbb{Z}^d afin d'en obtenir une partition en rectangles (voir la Figure 5).

Soit $p \in A^{\mathbb{U}_{\mathbf{k}}}$ un motif rectangulaire de support fini $\mathbb{U}_{\mathbf{k}} \subset \mathbb{Z}^d$. On dit que la substitution s est *compatible* avec le motif p (resp. la configuration x) si pour tous $\mathbf{i} = (\mathbf{i}_1, \dots, \mathbf{i}_d) \in \mathbb{U}_{\mathbf{k}}$ et $\mathbf{j} = (\mathbf{j}_1, \dots, \mathbf{j}_d) \in \mathbb{U}_{\mathbf{k}}$ (resp. $\mathbf{i} = (\mathbf{i}_1, \dots, \mathbf{i}_d) \in \mathbb{Z}^d$ et $\mathbf{j} = (\mathbf{j}_1, \dots, \mathbf{j}_d) \in \mathbb{Z}^d$) tels que $\mathbf{i}_l = \mathbf{j}_l$ pour un certain $l \in [1; d]$, on a $\mathbf{k}_l^s(p_{\mathbf{i}}) = \mathbf{k}_l^s(p_{\mathbf{j}})$. Étant donnée une substitution s compatible avec une configuration $x \in \mathbb{Z}^d$, on peut transformer le réseau \mathbb{Z}^d en un réseau non régulier grâce à la fonction $\phi^{(x,s)} = \phi^{(\mathbf{k}^s(x_{(n,\dots,n)}))_{n \in \mathbb{Z}}}$ (voir la Figure 6).

Si la substitution s est compatible avec la configuration x qui contient un motif p , la substitution s agit sur p et on obtient un motif $s(p)$ dont le support est

$$\text{supp}(s(p)) = \bigcup_{\mathbf{i} \in \text{supp}(p)} \mathbb{U}_{\mathbf{k}^s(p_{\mathbf{i}})} + \phi^{(x,s)}(\mathbf{i})$$

et tel que

$$\forall \mathbf{i} \in \text{supp}(p), \forall \mathbf{j} \in \text{supp}(s(p_{\mathbf{i}})), s(p)_{\phi^{(x,s)}(\mathbf{i})+\mathbf{j}} = s(p_{\mathbf{i}})_{\mathbf{j}}.$$

Ainsi la substitution s peut facilement s'étendre en une fonction sur les configurations $s : \begin{pmatrix} A^{\mathbb{Z}^d} & \rightarrow & A^{\mathbb{Z}^d} \\ x & \mapsto & s(x) \end{pmatrix}$ où la configuration $s(x)$ est définie comme

expliqué ci-dessus, à condition que la substitution s soit compatible avec la configuration $x \in A^{\mathbb{Z}^d}$.

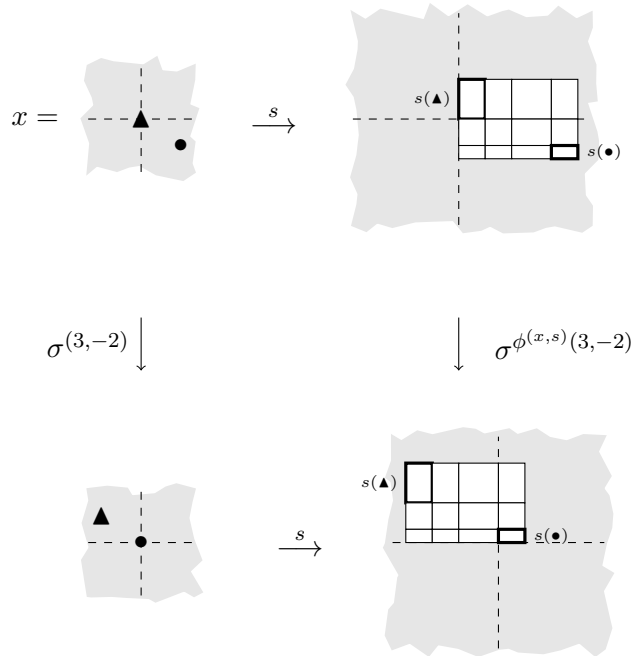


FIGURE 6: Si la substitution s est compatible avec la configuration x , alors s agit sur x et produit la configuration $s(x)$.

Exemple 2.1.1. Soit A l'alphabet à deux éléments $A = \{\circ, \bullet\}$ et s la substitution données par les règles suivantes :

$$\circ \mapsto \begin{array}{cc} \circ & \circ \\ \circ & \circ \end{array} \quad \text{et} \quad \bullet \mapsto \begin{array}{cc} \circ & \circ \\ \bullet & \circ \end{array} .$$

Dans cet exemple, le support de $s(\circ)$ et de $s(\bullet)$ sont les mêmes, donc s est compatible avec n'importe quel motif. Par exemple, s opère sur le motif p ci-dessous :

$$s : p = \begin{array}{ccc} \circ & \bullet & \bullet \\ \bullet & \circ & \circ \end{array} \mapsto s(p) = \begin{array}{ccc|ccc|ccc} \circ & \circ & & \circ & \circ & & \circ & \circ & \\ \circ & \circ & & \bullet & \circ & & \bullet & \circ & \\ \circ & \circ & & \circ & \circ & & \circ & \circ & \\ \bullet & \circ & & \circ & \circ & & \circ & \circ & \end{array} .$$

2.1.2 Composition de substitutions

Nous venons de voir comment une substitution peut agir sur un motif fini ou sur une configuration infinie. Supposons maintenant qu'il nous est donné un ensemble fini de substitutions, comment le faire agir sur un motif ou une configuration ?

Définissons d'abord la composition de deux substitutions. Soient s, s' deux substitutions. On dit que s' est *compatible* avec s si pour tout motif p compatible avec s , $s(p)$ est compatible avec s' . Si s' est compatible avec s , on peut alors définir la composition $s' \circ s$ de la manière suivante : le motif $s' \circ s(p)$ est obtenu en appliquant la substitution s' au motif $s(p)$. Pour une suite de substitutions $S_{[0;n]} = (s_0, \dots, s_n)$, la substitution $\widehat{S}_{[k;n]}$ est définie par induction de la manière suivante pour tout entier $k \leq n$:

$$\begin{cases} \widehat{S}_{[k;n]} = s_n & \text{si } k = n; \\ \widehat{S}_{[k;n]} = s_k \circ \widehat{S}_{[k+1;n]} & \text{si } k < n \text{ et si } s_k \text{ est compatible avec } \widehat{S}_{[k+1;n]}. \end{cases}$$

Notons que si jamais la condition de compatibilité n'est pas vérifiée à une des étapes de la définition de la substitution $\widehat{S}_{[k;n]}$, alors celle-ci n'est pas définie.

Considérons désormais \mathcal{S} un ensemble fini de substitutions de même dimension sur un même alphabet. Dans la suite on présente deux points de vue pour faire agir cet ensemble sur une configuration $x \in A^{\mathbb{Z}^d}$. Dans un premier temps, l'ensemble \mathcal{S} agit sur une configuration x au moyen d'une suite de substitutions $S = (s_i)_{i \in \mathbb{N}} \in \mathcal{S}^{\mathbb{N}}$, chaque substitution étant appliquée de manière uniforme à toutes les lettres d'une configuration (Partie 2.1.3). Mais l'ensemble \mathcal{S} peut également agir de manière non uniforme sur une configuration x , la substitution appliquée à une lettre de x dépendant cette fois de la position de la lettre (Partie 2.1.4).

2.1.3 Décalages S-adiques

Soit \mathcal{S} un ensemble fini de substitutions de dimension d sur le même alphabet A , et soit $S \in \mathcal{S}^{\mathbb{N}}$ une suite de substitutions. Partant d'une lettre de l'alphabet a , cette suite de substitutions opère sur a de la manière suivante. Lors de la $i^{\text{ème}}$ étape, la substitution s_i est appliqué à l'ensemble des lettres formant le motif $s_1 \circ \dots \circ s_i(a)$. Deux décalages, appelés décalages *S-adiques*, sont définis en se basant sur ce principe.

Dans une première approche combinatoire, on s'intéresse au langage des motifs produits par la suite S , appelés les S -motifs, que l'on interprète comme les motifs autorisés d'un décalage. Le *décalage S-adique local* engendré par la suite

de substitutions S , noté \mathbf{T}_S , est défini de la façon suivante :

$$\mathbf{T}_S = \left\{ x \in A^{\mathbb{Z}^d} : \forall p \sqsubset x, \exists a \in A, \exists n \in \mathbb{N}, p \sqsubset \widehat{S}_{[0;n]}(a) \right\}.$$

C'est donc l'ensemble des configurations telles que tout motif apparaissant dans la configuration apparaît également dans un S -motif.

L'autre approche, correspondant à une vision dynamique, consiste à faire agir la suite de substitutions S sur l'ensemble des configurations $A^{\mathbb{Z}^d}$. Le *décalage S -adique global* engendré par la suite de substitutions S , noté \mathbf{T}'_S , est alors :

$$\mathbf{T}'_S = \bigcup_{i \in \mathbb{Z}^d} \sigma^i \left(\left\{ x \in A^{\mathbb{Z}^d} : \forall n \in \mathbb{N}, \exists y \in A^{\mathbb{Z}^d}, \widehat{S}_{[0;n]}(y) = x \right\} \right).$$

Il s'agit de l'ensemble des configurations pour lesquelles on peut trouver, à une translation près, un antécédent par chacune des itérées de la substitution s .

Il est aisé de vérifier que si S est une suite de substitutions, alors $\mathbf{T}_S \subseteq \mathbf{T}'_S$.

Exemple 2.1.2. Soit s la substitution de l'exemple 2.1.1, on considère la suite $S = s^{\mathbb{N}}$. Alors le décalage substitutif local associé à cette substitution est :

$$\mathbf{T}_{\{s\}} = \{ \circ^{\mathbb{Z}^2} \},$$

tandis que son décalage substitutif global est :

$$\mathbf{T}'_s = \{ \circ^{\mathbb{Z}^2} \} \cup \{ \sigma^i(x_{\bullet}), i \in \mathbb{Z}^2 \},$$

où la configuration x_{\bullet} est telle que $x_{(i,j)} = \circ$ si $i \neq 0$ ou $j \neq 0$ et $x_{(0,0)} = \bullet$. Cette configuration n'appartient pas au décalage substitutif local $\mathbf{T}_{\{s\}}$ car le motif central x_{S_1} n'apparaît dans aucun des s -motifs.

2.1.4 Décalages substitutifs non déterministes

Soit \mathcal{S} un ensemble de substitutions ; il est possible de faire agir cet ensemble sur une configuration de manière non déterministe, ou plus précisément de manière non uniforme. Étant donné un motif $p \in A^{\mathbb{U}}$, on applique une substitution sur chacune de ses lettres, qui peut être différente selon la position de la lettre dans le motif.

Si $\mathbb{U} \subset \mathbb{Z}^d$ est un support fini, on appelle *motif de substitution* un élément $\mathbf{s} \in \mathcal{S}^{\mathbb{U}}$. On dit que le motif de substitution $\mathbf{s} \in \mathcal{S}^{\mathbb{U}}$ est *compatible* avec un motif $p \in A^{\mathbb{U}}$ qui apparaît dans $x \in A^{\mathbb{Z}^d}$ si pour tout $\mathbf{i} = (i_1, \dots, i_d) \in \mathbb{U}$ et $\mathbf{j} = (j_1, \dots, j_d) \in \mathbb{U}$ tel que $i_l = j_l$ pour un certain $l \in [1; d]$, on a $\mathbf{k} =_{i_l}^{s_i} (p_i) = \mathbf{k}_l^{s_j} (p_j)$.

Si le motif de substitution $\mathbf{s} \in \mathcal{S}^{\mathbb{U}_k}$ est compatible avec un motif $p \in A^{\mathbb{U}_k}$, il agit sur p et on obtient le motif $\mathbf{s}(p)$, dont le support est $\text{supp}(\mathbf{s}(p)) = \bigcup_{i \in \text{supp}(p)} \mathbb{U}_{\mathbf{k}^{s_i}(p_i)} + \phi^{(x, \mathbf{s})}(\mathbf{i})$, défini par :

$$\forall \mathbf{i} \in \text{supp}(p), \forall \mathbf{j} \in \text{supp}(\mathbf{s}_i(p_i)), \mathbf{s}(p)_{\phi^{(x, \mathbf{s})}(\mathbf{i}) + \mathbf{j}} = \mathbf{s}_i(p_i)_{\mathbf{j}}.$$

Exemple 2.1.3. Soit $\mathcal{S} = \{s_1, s_2, s_3, s_4\}$ un ensemble de substitutions de dimension 2, définies sur l'alphabet $A = \{\circ, \bullet\}$, et dont les règles sont les suivantes :

$$\begin{aligned} s_1 : \circ &\mapsto \begin{array}{cc} \circ & \circ \\ \circ & \circ \end{array} \quad \text{et} \quad \bullet &\mapsto \begin{array}{cc} \circ & \circ \\ \bullet & \circ \end{array}, \quad s_2 : \circ &\mapsto \begin{array}{ccc} \circ & \bullet & \circ \\ \circ & \bullet & \circ \end{array} \quad \text{et} \quad \bullet &\mapsto \begin{array}{ccc} \circ & \circ & \circ \\ \bullet & \circ & \circ \end{array} \\ s_3 : \circ &\mapsto \begin{array}{cc} \circ & \circ \\ \bullet & \circ \end{array} \quad \text{et} \quad \bullet &\mapsto \begin{array}{cc} \circ & \circ \\ \circ & \bullet \end{array}, \quad s_4 : \circ &\mapsto \begin{array}{ccc} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{array} \quad \text{et} \quad \bullet &\mapsto \begin{array}{ccc} \circ & \circ & \circ \\ \bullet & \bullet & \bullet \end{array}. \end{aligned}$$

Considérons le motif p dessiné ci-dessous. Alors les motifs de substitution \mathbf{s} et \mathbf{s}' sont compatibles avec p et définissent deux motifs $\mathbf{s}(p)$ et $\mathbf{s}'(p)$, tandis que le motif de substitution \mathbf{s}'' n'est pas compatible avec p .

$$p = \begin{array}{cccc} \circ & \bullet & \bullet & \bullet \\ \bullet & \bullet & \circ & \circ \end{array}$$

$$\begin{aligned} \mathbf{s} &= \begin{array}{cc} s_1 & s_1 \\ s_3 & s_3 \end{array} \begin{array}{cc} s_1 & s_2 \\ s_3 & s_4 \end{array} \begin{array}{cc} s_1 & s_1 \\ s_3 & s_3 \end{array}, \quad \mathbf{s}' = \begin{array}{cc} s_1 & s_1 \\ s_3 & s_3 \end{array} \begin{array}{cc} s_1 & s_1 \\ s_3 & s_3 \end{array}, \quad \mathbf{s}'' = \begin{array}{cc} s_1 & s_1 \\ s_3 & s_3 \end{array} \begin{array}{cc} s_2 & s_1 \\ s_4 & s_1 \end{array} \\ \mathbf{s}(p) &= \begin{array}{cc|cc|ccc|cc} \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \bullet & \circ & \circ & \bullet \\ \circ & \circ & \circ & \circ & \bullet & \bullet & \bullet & \circ \\ \circ & \bullet & \circ & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \circ & \circ & \circ & \bullet \end{array}, \quad \mathbf{s}'(p) = \begin{array}{cc|cc|ccc|cc} \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \bullet & \circ & \circ & \circ & \circ & \circ & \circ & \bullet \\ \circ & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \circ & \circ & \circ & \circ & \circ & \bullet & \circ & \circ \\ \bullet & \circ & \bullet & \circ & \circ & \bullet & \circ & \circ \end{array} \end{aligned}$$

L'ensemble des \mathcal{S} -motifs est défini par induction. Un \mathcal{S} -motif de niveau 0 est une lettre de l'alphabet A , et p est un \mathcal{S} -motif de niveau $n+1$ s'il existe un \mathcal{S} -motif $p' \in A^{\mathbb{U}}$ de niveau n et un motif de substitution $\mathbf{s} \in \mathbb{S}^{\mathbb{U}}$ compatible avec p' tel que $\mathbf{s}(p') = p$. Le support d'un \mathcal{S} -motif est donc toujours rectangulaire. Les \mathcal{S} -motifs permettent de définir $\mathbf{T}_{\mathcal{S}}$, le *décalage substitutif local* engendré par l'ensemble de substitutions \mathcal{S} :

$$\mathbf{T}_{\mathcal{S}} = \left\{ x \in A^{\mathbb{Z}^d} : \forall p \sqsubset x, p \text{ est un sous-motif d'un } \mathcal{S}\text{-motif} \right\}.$$

Remarque. Pour toute substitution s , on a $\mathbf{T}_{\{s\}} = \mathbf{T}_{s^{\mathbb{N}}}$.

Supposons que $\mathbf{s} \in \mathcal{S}^{\mathbb{Z}^d}$ est un motif de substitution infini compatible avec une configuration $x \in A^{\mathbb{Z}^d}$. On note $\mathbf{s}(x)$ la configuration dans $A^{\mathbb{Z}^d}$ obtenue en

appliquant la substitution \mathbf{s}_i sur la lettre x_i pour tout $i \in \mathbb{Z}^d$, et en accolant les motifs $s_i(x_i)$ ainsi obtenus. On définit $\mathbf{T}'_{\mathcal{S}}$ le *décalage substitutif global* engendré par l'ensemble de substitutions \mathcal{S} par :

$$\mathbf{T}'_{\mathcal{S}} = \bigcup_{i \in \mathbb{Z}^d} \sigma^i \left(\left\{ x \in A^{\mathbb{Z}^d} : \forall n \in \mathbb{N}, \exists y \in A^{\mathbb{Z}^d}, \exists (\mathbf{s}_0, \dots, \mathbf{s}_{n-1}) \in (\mathcal{S}^{\mathbb{Z}^d})^n, \mathbf{s}_0 \circ \dots \circ \mathbf{s}_{n-1}(y) = x \right\} \right).$$

Remarque. Les deux inclusions $\mathbf{T}_S \subseteq \mathbf{T}_{\mathcal{S}}$ et $\mathbf{T}'_S \subseteq \mathbf{T}'_{\mathcal{S}}$ sont vérifiées pour toute suite de substitutions S sur un ensemble de substitutions \mathcal{S} .

2.1.5 Substitutions non-déterministes et théorème de Mozes

Dans l'article [Moz89], Mozes étudie les substitutions multidimensionnelles non déterministes, et montre qu'à condition qu'une substitution vérifie une certaine propriété, le décalage qu'elle engendre est sofique.

Toutes les substitutions considérées jusqu'ici sont déterministes, puisque leur ensemble de règles est donné par une fonction. Cependant le formalisme présenté avec les ensembles de substitutions \mathcal{S} englobe aussi ces substitutions non déterministes. Étant donnée s une substitution non déterministe, si une lettre $a \in A$ a deux images possibles p_1 et p_2 , on remplace la substitution s par deux substitutions s_1 et s_2 , où s_1 possède les mêmes règles de substitution que s mise à part la règle $a \rightarrow p_2$, et s_2 possède les mêmes règles de substitution que s mise à part la règle $a \rightarrow p_1$. En répétant ce procédé, on peut transformer une substitution non déterministe s en un ensemble de substitutions déterministes \mathcal{S} de sorte que le décalage (Ω, \mathbb{Z}^2) défini par Mozes corresponde exactement au décalage $\mathbf{T}_{\mathcal{S}}$.

Théorème 2.1.1 ([Moz89]). *Soit \mathcal{S} un ensemble de substitutions multidimensionnelles non dégénérées possédant la propriété A . Alors le décalage $\mathbf{T}_{\mathcal{S}}$ est sofique.*

On dit qu'un ensemble de substitutions \mathcal{S} est *de type A* , ou *possède la propriété A* , s'il vérifie la condition définie ci-après. Soit p un \mathcal{S} -motif et l un sous-motif de taille 2×2 apparaissant dans p . Supposons qu'il existe une suite de motifs de substitution $\mathbf{s}_1, \dots, \mathbf{s}_n$ compatible avec le motif l , qui produit une suite de motifs $l_0 = l, l_1 = \mathbf{s}_1(l_0), \dots, l_n = \mathbf{s}_n(l_{n-1})$. Alors il est possible de trouver une suite de motifs de substitution $\mathbf{s}'_1, \dots, \mathbf{s}'_n$ compatible avec le motif p telle que les motifs dérivant de l dans $p_0 = p, p_1 = \mathbf{s}'_1(p_0), \dots, p_n = \mathbf{s}'_n(p_{n-1})$ sont exactement les l_0, l_1, \dots, l_n (voir la Figure 7).

Cette propriété A n'est *a priori* pas triviale. Il est en effet possible que la suite de motifs de substitution choisie pour l ne soit pas compatible avec le motif p . Dans ce cas il est quand même possible de trouver une autre suite de

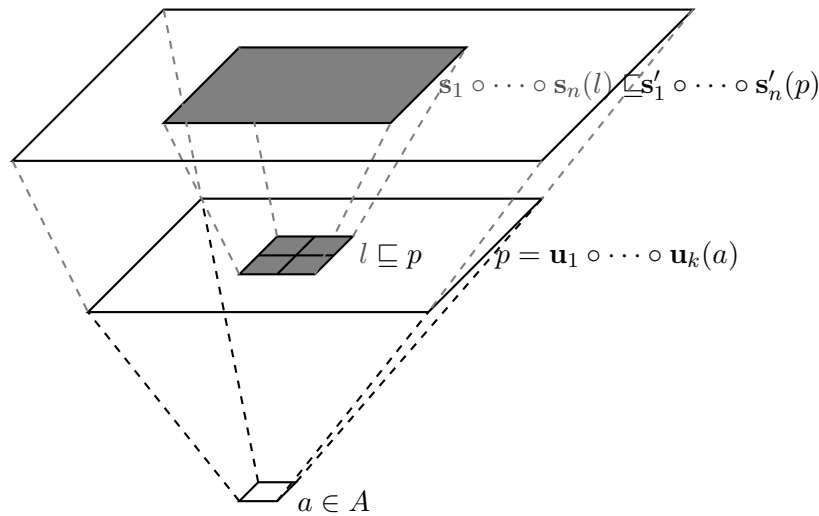


FIGURE 7: Un ensemble de substitutions \mathcal{S} qui possède la propriété A .

motifs de substitution compatible avec p et telle que les motifs dérivant de l sont bien ceux recherchés.

Remarque. La propriété A est tout de même vérifiée par bon nombre d'ensembles de substitutions. Par exemple tout ensemble de substitutions \mathcal{S} dans lequel l'image d'une lettre par une substitution $s \in \mathcal{S}$ ne dépend que de la substitution s possède la propriété A . De plus, si l'ensemble \mathcal{S} est réduit à une unique substitution déterministe, alors \mathcal{S} possède la propriété A .

Remarque. Si $S \in \mathcal{S}^{\mathbb{N}}$ est une suite de substitutions, on sait que $\mathbf{T}_S \subset \mathbf{T}_{\mathcal{S}}$ qui est sofique, mais il n'y a *a priori* aucune raison évidente pour que \mathbf{T}_S le soit également.

Il est en fait possible d'adapter la preuve de Mozes pour montrer un résultat similaire pour le décalage $\mathbf{T}'_{\mathcal{S}}$. Il n'est d'ailleurs plus nécessaire dans ce cas de considérer un ensemble de substitutions possédant la propriété A .

Corollaire 2.1.2. [AS11] Soit \mathcal{S} un ensemble de substitutions multidimensionnelles déterministes. Alors le décalage $\mathbf{T}'_{\mathcal{S}}$ est sofique.

Idée de la démonstration Nous donnons les principales idées pour adapter la preuve originale du Théorème 2.1.1, et ainsi obtenir un schéma de preuve du Corollaire 2.1.2. Soit \mathcal{S} un ensemble de substitutions qui possède la propriété A .

Mozes construit un décalage sofique Σ tel que $\mathbf{T}_{\mathcal{S}}$ est un facteur de Σ . Le décalage Σ contient une grille qui assure qu'une configuration x appartient au décalage sofique Σ si et seulement si on peut trouver, pour tout $n \in \mathbb{N}$, une suite de motifs de substitutions infinis $\mathbf{s}_0, \dots, \mathbf{s}_{n-1} \in \mathcal{S}^{\mathbb{Z}^d}$ et une configuration y_n tels que $\mathbf{s}_0 \circ \dots \circ \mathbf{s}_{n-1}(y_n) = x$. De plus dans Σ tous les y_n sont codés à l'intérieur d'une structure hiérarchique. On appelle Q l'ensemble des motifs de taille 2×2 qui apparaissent dans un \mathcal{S} -motif. En plus de ce réseau permettant de dé-substituer la configuration x autant qu'on le souhaite, on ajoute la condition suivante : tout motif de taille 2×2 qui apparaît dans une configuration y_n appartient à l'ensemble Q . Cette condition assure la validité de la construction : étant donné une configuration $x \in \mathbf{T}_{\mathcal{S}}$ il est facile de construire une configuration y de Σ qui code x et toutes ses pré-images. Réciproquement étant donné un motif p qui apparaît dans une configuration $x \in \Sigma$, on peut trouver une suite de motifs finis de substitution $(\mathbf{s}_0, \dots, \mathbf{s}_{n-1})$ telle que p apparaît dans $\mathbf{s}_0 \circ \dots \circ \mathbf{s}_{n-1}(\tilde{p})$, où \tilde{p} est soit réduit à une lettre, soit un motif de taille 2×1 , 1×2 ou 2×2 . Si \tilde{p} est une lettre alors on en déduit immédiatement que p apparaît dans un \mathcal{S} -motif. Sinon, \tilde{p} apparaît dans un motif de taille 2×2 qui apparaît lui-même dans un \mathcal{S} -motif (grâce à la condition Q), et la propriété A assure alors que le motif p apparaît aussi dans un \mathcal{S} -motif (voir la Figure 7). Ainsi l'utilisation conjointe de la propriété A et de la condition Q assure que tout motif apparaissant dans x apparaît aussi dans un \mathcal{S} -motif.

La différence entre les décalages $\mathbf{T}_{\mathcal{S}}$ et $\mathbf{T}'_{\mathcal{S}}$ s'exprime essentiellement avec les \mathcal{S} -motifs. Toute configuration x appartenant à l'un de ces deux décalages doit posséder un antécédent d'ordre arbitraire par \mathcal{S} , ce que la construction de Mozes nous assure, mais on impose à tout motif apparaissant dans x d'être un \mathcal{S} -motif uniquement pour le décalage $\mathbf{T}_{\mathcal{S}}$. Ainsi pour adapter la preuve de Mozes au décalage $\mathbf{T}'_{\mathcal{S}}$, la propriété A n'est plus nécessaire, et on remplace l'ensemble Q par l'ensemble de tous les motifs de taille 2×2 . Le décalage $\mathbf{T}'_{\mathcal{S}}$ est un facteur du décalage sofique obtenu, ce qui prouve le corollaire.

2.2 Opérations et simulation

Cette partie étudie l'ensemble des décalages sur \mathbb{Z}^d via l'action d'opérations sur cet ensemble. Dans ce but, une notion de simulation d'un décalage par un autre est présentée.

2.2.1 Simulation et classes stables

Une *opération* op sur l'ensemble des décalages est une fonction $op : \mathcal{E} \rightarrow \mathcal{E}$ ou $op : \mathcal{E} \times \mathcal{E} \rightarrow \mathcal{E}$. Une *classe d'opérations* sur l'ensemble des décalages est un ensemble de fonctions $op_\ell : \mathcal{E} \rightarrow \mathcal{E}$ ou $op_\ell : \mathcal{E} \times \mathcal{E} \rightarrow \mathcal{E}$ dépendant d'un paramètre ℓ . Dans ces définitions rien n'impose qu'un décalage et son image soient définis sur le même alphabet, ni qu'ils aient la même dimension. Dans la suite, par abus de langage, on utilisera parfois le terme opération pour désigner une classe d'opérations.

Supposons que nous soit donné un ensemble Op d'opérations et de classes d'opérations sur les décalages. Deux notions peuvent naturellement venir à l'esprit lorsqu'il s'agit d'étudier de quelle façon cet ensemble agit sur les décalages.

La première est la notion de simulation. Étant donné un ensemble Op d'opérations ou de classes d'opérations agissant sur les décalages, on dit qu'un décalage \mathbf{T} *simule* un décalage \mathbf{T}' si on peut obtenir \mathbf{T}' en partant de \mathbf{T} et en appliquant un nombre fini d'opérations choisies parmi Op . On note $\mathbf{T}' \leq_{Op} \mathbf{T}$. Ainsi $Cl_{Op}(\mathbf{T}) = \{\mathbf{T}' : \mathbf{T}' \leq_{Op} \mathbf{T}\}$.

La seconde est la notion de stabilité. Soit \mathcal{U} un ensemble de décalages (ou classe de décalages). La *clôture* de \mathcal{U} pour l'ensemble d'opérations Op , que l'on note $Cl_{Op}(\mathcal{U})$, est le plus petit ensemble stable par Op qui contient \mathcal{U} . Une classe de décalages \mathcal{U} est *stable* pour l'ensemble d'opérations Op si $Cl_{Op}(\mathcal{U}) = \mathcal{U}$.

2.2.2 Exemples d'opérations et résultats classiques de simulation

Cette partie définit quatre opérations sur les décalages, présente des classes stables évidentes et propose quelques exemples d'application de ces opérations.

Opération type fini (TF) : Cette classe d'opérations consiste à ajouter un nombre fini de motifs interdits au décalage sur lequel une opération **TF** est appliquée. Formellement, si A est un alphabet, $P \subseteq \mathcal{E}_A^d$ un ensemble fini de motifs et $\mathbf{T} \subseteq A^{\mathbb{Z}^d}$ un décalage, on définit l'opération **TF** avec ensemble de motifs P par :

$$\mathbf{TF}_P(\mathbf{T}) = \mathbf{T}_{P \cup P'},$$

où l'ensemble P' , donné par la Proposition 1.2.3, est un ensemble de motifs interdits qui définit le décalage $\mathbf{T} : \mathbf{T} = \mathbf{T}_{P'}$.

Le décalage $\mathbf{TF}_P(\mathbf{T})$ peut être vide si jamais l'ensemble P est trop restrictif. Par définition des décalages de type fini, on a :

$$Cl_{\mathbf{TF}}(\mathcal{FS}) = \mathcal{SFT} \text{ et } Cl_{\mathbf{TF}}(\mathcal{FS}) = \mathcal{SFT}.$$

Opération produit (Prod) : Étant données n configurations $x_i \subseteq A_i^{\mathbb{Z}^d}$ pour $i \in \{1, \dots, n\}$ de même dimension, on définit leur produit :

$$x_1 \times \cdots \times x_n = (x_1, \dots, x_n) \in (A_1 \times \cdots \times A_n)^{\mathbb{Z}^d}.$$

Étant donnés n décalages $\mathbf{T}_i \subseteq A_i^{\mathbb{Z}^d}$ pour $i \in \{1, \dots, n\}$, on définit le décalage produit comme l'ensemble des configurations

$$\mathbf{Prod}(\mathbf{T}_1, \dots, \mathbf{T}_n) = \{x_1 \times \cdots \times x_n \mid \forall i = 1..n, x_i \in \mathbf{T}_i\} \subseteq (A_1 \times \cdots \times A_n)^{\mathbb{Z}^d}.$$

Ainsi, pour $p_i \in A_i^{\mathbb{S}}$, le motif produit $p_1 \times \cdots \times p_n \in (A_1 \times \cdots \times A_n)^{\mathbb{S}}$ est interdit dans le décalage $\mathbf{Prod}(\mathbf{T}_1, \dots, \mathbf{T}_n)$ si et seulement si un des p_i est interdit dans \mathbf{T}_i .

Cette opération consiste donc à superposer les configurations des décalages \mathbf{T}_i . Elle permet également de faire grossir l'alphabet d'un décalage. Par exemple si \mathbf{T} est un décalage sur l'alphabet A , alors le décalage produit $\mathbf{T} \times B^{\mathbb{Z}^d}$ est le décalage inclus dans $(A \times B)^{\mathbb{Z}^d}$ défini par les même motifs interdits que \mathbf{T} .

Exemple 2.2.1. Soit \mathbf{T}_1 le décalage de type fini de dimension 2 défini sur l'alphabet $\{a, b, c\}$, donné par les motifs interdits suivants :

$$F_1 = \left\{ \begin{array}{|c|c|} \hline * & a \\ \hline a & * \\ \hline \end{array}, \begin{array}{|c|c|} \hline * & b \\ \hline b & * \\ \hline \end{array}, \begin{array}{|c|c|} \hline * & c \\ \hline c & * \\ \hline \end{array} \right\};$$

dans lesquels le symbole $*$ remplace n'importe quelle lettre de l'alphabet.

Soit \mathbf{T}_2 le décalage de type fini, toujours en dimension 2, défini cette fois sur l'alphabet $\{\blacksquare, \blacksquare\}$, dont les motifs interdits sont :

$$F_2 = \left\{ \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \end{array}, \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \end{array}, \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \end{array}, \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \end{array} \right\}.$$

Alors le décalage produit $\mathbf{T}_1 \times \mathbf{T}_2$ est un décalage de dimension 2 défini sur l'alphabet produit $\{a, b, c, a, b, c\}$, dont les motifs interdits sont de la forme :

$$\left\{ \begin{array}{|c|c|} \hline * & a \\ \hline a & * \\ \hline \end{array}, \begin{array}{|c|c|} \hline * & a \\ \hline a & * \\ \hline \end{array}, \dots, \begin{array}{|c|c|} \hline * & * \\ \hline * & * \\ \hline \end{array}, \dots \right\};$$

plus précisément $p = p_1 \times p_2 \in \{a, b, c, a, b, c\}^{[0,1] \times [0,1]}$ est interdit si et seulement si $p_1 \in F_1$ ou $p_2 \in F_2$.

Dans l'exemple 2.2.1, le produit de deux décalages de type fini est lui aussi de type fini. C'est en fait le cas pour n'importe quel produit de SFT, ce que l'on exprime en terme de clôture de classe par

$$Cl_{\mathbf{Prod}}(\mathcal{SFT}) = \mathcal{SFT}.$$

Une autre classe stable évidente par cette opération est la classe des décalages pleins :

$$Cl_{\mathbf{Prod}}(\mathcal{FS}) = \mathcal{FS}.$$

Opération facteur (Fact) : Cette classe d'opérations permet de transformer l'alphabet d'un décalage par des modifications locales, et consiste à appliquer une fonction de bloc à ce dernier (voir la Définition 6 page 21). Soient A et B deux alphabets finis, $\pi : A^{\mathbb{Z}^d} \rightarrow B^{\mathbb{Z}^d}$ une fonction de bloc et $\mathbf{T} \subseteq A^{\mathbb{Z}^d}$ un décalage. On définit :

$$\mathbf{Fact}_\pi(\mathbf{T}) := \pi(\mathbf{T}).$$

L'exemple 1.3.2 montre que la classe des décalages de type fini n'est pas stable par l'opération **Fact** :

$$\mathcal{SFT} \not\subseteq \mathcal{Cl}_{\mathbf{Fact}}(\mathcal{SFT}).$$

Par définition des décalages sofiques, on a :

$$\mathcal{Cl}_{\mathbf{Fact}}(\mathcal{SFT}) = \mathit{Sofique}.$$

Opération extension (SE) : Cette opération permet d'augmenter la dimension d'un décalage. Soit $d, d' \in \mathbb{N}^*$, et soient \mathbb{G} et \mathbb{G}' deux sous-groupes de $\mathbb{Z}^{d+d'}$ tels que \mathbb{G} est isomorphe à \mathbb{Z}^d et $\mathbb{G} \oplus \mathbb{G}' = \mathbb{Z}^{d+d'}$. Soit $\mathbf{T} \subseteq A^{\mathbb{Z}^d}$ un décalage de dimension d . On définit l'extension de \mathbf{T} comme le décalage :

$$\mathbf{SE}_{\mathbb{G}, \mathbb{G}'}(\mathbf{T}) = \left\{ x \in A^{\mathbb{Z}^{d+d'}} : \forall i \in \mathbb{G}', x_{i+\mathbb{G}} \in \mathbf{T} \right\}.$$

Il est facile de vérifier que cette opération laisse stable la classe des décalages de type fini $\mathcal{Cl}_{\mathbf{SE}}(\mathcal{SFT}) = \mathcal{SFT}$.

2.2.3 La sous-action projective

Cette partie est dédiée à l'opération de sous-action projective. Cette opération consiste à ne regarder un décalage que selon un sous-groupe de \mathbb{Z}^d , en laissant de côté le reste du décalage.

Définition 12. Soit \mathbb{G} un sous-groupe de \mathbb{Z}^d librement engendré par $u_1, u_2, \dots, u_{d'}$ ($d' \leq d$). Étant donné $\mathbf{T} \subseteq A^{\mathbb{Z}^d}$ un décalage, on définit la *sous-action projective de \mathbf{T} sur \mathbb{G}* (ou encore la *projection de \mathbf{T} sur \mathbb{G}*) par :

$$\mathbf{SA}_{\mathbb{G}}(\mathbf{T}) = \left\{ y \in A^{\mathbb{Z}^{d'}} : \exists x \in \mathbf{T} \text{ tel que } \forall i_1, \dots, i_{d'} \in \mathbb{Z}^d, y_{i_1, \dots, i_{d'}} = x_{i_1 u_1 + \dots + i_{d'} u_{d'}} \right\}.$$

On peut facilement vérifier que $\mathcal{Cl}_{\mathbf{SA}}(\mathcal{SFT}) \neq \mathcal{SFT}$ (voir l'exemple 2.2.2) et que $\mathcal{Cl}_{\mathbf{SA}}(\mathcal{SFT}) \neq \mathit{Sofique}$ et $\mathcal{Cl}_{\mathbf{SA}}(\mathit{Sofique}) \neq \mathit{Sofique}$ (voir les exemples 2.2.2 et 2.2.3).

Exemple 2.2.2. On se place sur \mathbb{Z}^2 et on construit un décalage de type fini $\mathbf{T} \subset \{\blacksquare, \color{orange}\square, \color{pink}\square\}^{\mathbb{Z}^2}$ tel que la sous-action projective de \mathbf{T} sur le sous-groupe $\Delta = \{(x, y) \in \mathbb{Z}^2 : y = x\} \subseteq \mathbb{Z}^2$ n'est pas un décalage de type fini. Dans cet exemple le décalage qui apparaît le long de Δ est

$$\{x \in \{\color{green}\square, \color{orange}\square, \color{pink}\square\}^{\mathbb{Z}} : \text{les blocs de } \color{green}\square \text{ consécutifs sont de longueur paire}\}.$$

L'ensemble des motifs autorisés \overline{F} est défini comme l'ensemble des motifs de taille 4×4 décrit dans la Figure 8.

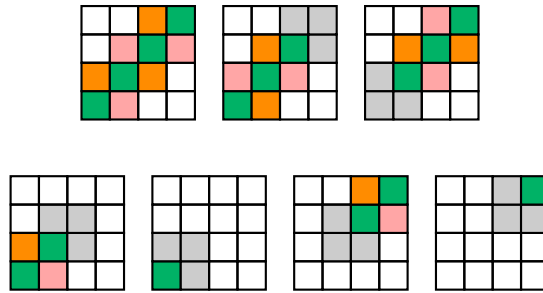


FIGURE 8: Les motifs autorisés du décalage de type fini \mathbf{T} .

L'alternance de $\color{orange}\square$ et de $\color{pink}\square$ le long des diagonales de $\color{green}\square$ permet de contrôler la parité des longueurs des blocs de $\color{green}\square$ consécutifs (voir la Figure 9).

Soit F l'ensemble de motifs de taille 4×4 qui ne sont pas dans \overline{F} . Alors en appelant \mathbf{T} le décalage défini par l'ensemble de motifs interdits F , on a :

$$\mathbf{SA}_{\Delta}(\mathbf{T}) = \{x \in \{\color{green}\square, \color{orange}\square, \color{pink}\square\}^{\mathbb{Z}} : \text{les blocs de } \color{green}\square \text{ consécutifs sont de longueur paire}\}$$

qui n'est pas un décalage de type fini (voir l'exemple 1.3.2).

Exemple 2.2.3. Le décalage de type fini explicité dans l'exemple 2.2.2 est sofique, mais il est possible d'obtenir des décalages non soifiques comme sous-action de SFT. On présente ici un décalage de type fini \mathbf{T} tel que la sous-action $\mathbf{SA}_{\Delta}(\mathbf{T})$ selon la diagonale $\Delta\{(x, y) \in \mathbb{Z}^2 : y = x\}$ n'est pas sofique. En dimension 1, les décalages soifiques sont exactement ceux dont le langage est régulier [LM95]. Le langage $\{a^n b^n : n \in \mathbb{N}\}$ est l'exemple classique de langage non régulier, et c'est ce langage que l'on souhaite faire apparaître le long de Δ .

L'alphabet du décalage \mathbf{T} est $A = \{\square, \color{pink}\square, \color{red}\square, \color{purple}\square, \color{blue}\square, \color{yellow}\square, \color{green}\square, \color{orange}\square\}$, et on définit le décalage $\mathbf{T} \subset A^{\mathbb{Z}^2}$ de façon à obtenir, sur un fond de symboles \square , des îlots de la forme :

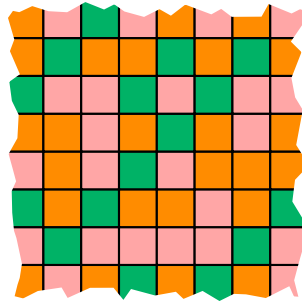
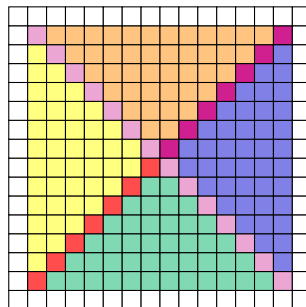
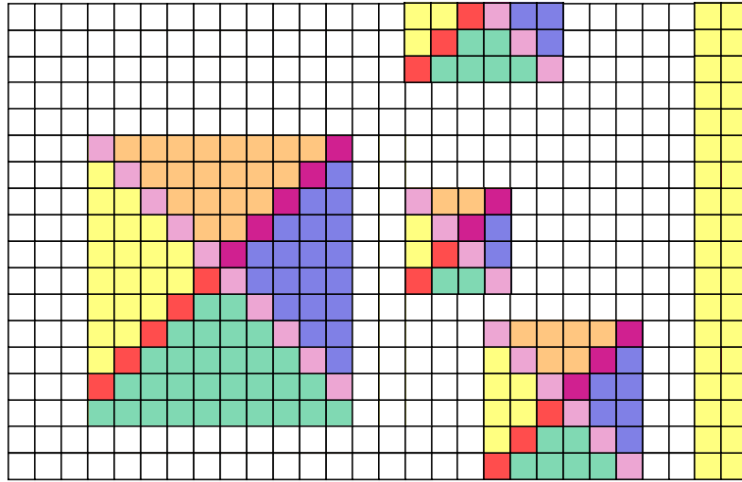


FIGURE 9: Une partie d'une configuration du décalage de type fini \mathbf{T} , dont la sous-action $\mathbf{SA}_\Delta(\mathbf{T})$ est un décalage sofique propre.



Ainsi les seules configurations possibles sont formés d'îlots de ce type, avec éventuellement un translaté d'un (ou deux s'ils sont compatibles) des demi-plans suivants : $\square^{N \times \mathbb{Z}}$, $\square^{-N \times \mathbb{Z}}$, $\square^{\mathbb{Z} \times N}$, $\square^{\mathbb{Z} \times (-N)}$:



Soit Π la 1-fonction de bloc définie localement par $\pi(x) = 0$ pour $x \in \{\square, \text{pink}, \text{blue}, \text{yellow}, \text{green}, \text{orange}\}$ et $\pi(\text{red}) = a, \pi(\text{magenta}) = b$. Si le décalage $\mathbf{SA}_\Delta(\mathbf{T})$ était sofique, alors $\Pi(\mathbf{SA}_\Delta(\mathbf{T}))$ le serait aussi par stabilité de la classe des soifiques par fonction de bloc. Mais ce n'est pas le cas, puisque

$$\Pi(\mathbf{SA}_\Delta(\mathbf{T})) = \mathbf{T}_{\{ba, a0, 0b\} \cup \{0a^m b^n 0 \mid m \neq n\}}.$$

Ces deux exemples montrent que ni la classe des SFT ni celle des décalages soifiques n'est stable par sous-action. On peut par contre montrer que la classe des décalages effectifs est une classe stable par sous-action.

Proposition 2.2.1. [AS09] *Sur \mathbb{Z}^d la classe des décalages récursivement énumérables est stable par sous-action projective :*

$$Cl_{\mathbf{SA}}(\mathcal{RE}) = \mathcal{RE}.$$

Démonstration. Soit $\mathbf{T} \subseteq A^{\mathbb{Z}^d}$ un décalage effectif donné par un ensemble de motifs interdits F récursivement énumérable. Soit \mathbb{G} un sous-groupe de \mathbb{Z}^d engendré par les vecteurs $u_1, \dots, u_d \in \mathbb{Z}^d$. Le principe de la démonstration est le suivant : à partir de l'ensemble F on construit un autre ensemble récursivement énumérable F' tel que $\mathbf{SA}_{\mathbb{G}}(\mathbf{T}) = \mathbf{T}_{F'}$. L'ensemble F' sera le complémentaire du langage du décalage $\mathbf{SA}_{\mathbb{G}}(\mathbf{T})$. On note par $\mathbf{T}' \subseteq A^{\mathbb{Z}^d}$ le décalage $\mathbf{SA}_{\mathbb{G}}(\mathbf{T})$.

Soit p' un motif dans $A^{\mathbb{S}_n^{d'}}$. On appelle $\Phi(p')$ l'ensemble des motifs élémentaires de $A^{\mathbb{Z}^d}$ qui contiennent le motif p . Soit F' l'ensemble suivant :

$$F' = \bigcup_{n \in \mathbb{N}} \{p' \in A^{\mathbb{S}_n^{d'}} : \forall p \in \Phi(p'), \exists \tilde{p} \in F \text{ tel que } \tilde{p} \sqsubseteq p\}.$$

Il suffit de montrer que F' est récursivement énumérable et que $\mathbf{T}' = \mathbf{T}_{F'}$.

Lemme 2.2.2. *L'ensemble de motifs F' est récursivement énumérable.*

Démonstration du Lemme 2.2.2. Comme F' est récursivement énumérable, il existe une machine de Turing \mathcal{M} dont le domaine est F' .

Soit \mathcal{M}_{aux} une machine de Turing auxiliaire dont le comportement sur un motif d'entrée p est le suivant : la machine \mathcal{M}_{aux} énumère tous les sous-motifs de p . Ils sont en nombre fini, et on les note p_1, \dots, p_n . Puis la machine \mathcal{M}_{aux} simule le comportement de la machine \mathcal{M} sur chacun des sous-motifs p_i , en consacrant à tour de rôle une étape de calcul à chacun des motifs. Dès que la machine \mathcal{M} s'arrête sur un des sous-motifs p_i , la machine \mathcal{M}_{aux} s'arrête aussi. Ainsi, la machine \mathcal{M}_{aux} s'arrête sur un motif p si et seulement si p possède un sous-motif dans l'ensemble F' .

On construit ensuite une machine de Turing \mathcal{M}' qui s'arrête sur un motif d'entrée p' si et seulement si $p' \in F'$. Soit p' un motif dans $A^{\mathbb{S}_n^{d'}}$. Sur le motif d'entrée p' , la machine \mathcal{M}' fonctionne de la façon suivante : elle énumère tous les supports élémentaires $\mathbb{S}_{n+1}^{d'}, \mathbb{S}_{n+2}^{d'}, \dots$ qui contiennent le support du motif p' . On appelle cette énumération $(\text{supp}_i)_{i \in \mathbb{N}}$. La machine \mathcal{M}' effectue ensuite les calculs suivants pour chaque support supp_i , en consacrant à tour de rôle une étape de calcul à chaque supp_i :

- soit C_i l'ensemble fini $A^{\text{supp}_i} \cap \Phi(p')$. On le note $C_i = \{p_1^{(i)}, \dots, p_{k_i}^{(i)}\}$;
- sur chaque $p_k^{(i)}$ à tour de rôle, la machine \mathcal{M}' simule la machine \mathcal{M}_{aux} .

De cette façon la machine \mathcal{M}' s'arrête sur un motif d'entrée p' si et seulement s'il existe un support supp_i contenant $\text{supp}(p')$ tel que pour tout motif $p \in \Phi(p')$ de support supp_i , le motif p contient un motif interdit de F' . Ceci correspond exactement à la définition de l'ensemble F' , qui est donc récursivement énumérable. \square

Lemme 2.2.3. $\mathbf{SA}_{\mathbb{G}}(\mathbf{T}) = \mathbf{T}_{F'}$

Démonstration du Lemme 2.2.3. • $\mathbf{SA}_{\mathbb{G}}(\mathbf{T}) \subseteq \mathbf{T}_{F'}$

Soit $y \in \mathbf{SA}_{\mathbb{G}}(\mathbf{T})$. Alors il existe une configuration $x \in \mathbf{T}$ telle que pour toute position $i = (i_1, \dots, i_{d'}) \in \mathbb{Z}^{d'}$, $y_i = x_{i_1 u_1 + \dots + i_{d'} u_{d'}}$. Soit p' un motif qui apparaît dans la configuration y . Si p' était dans l'ensemble F' , alors tout motif dans C_i , qui est le même ensemble que celui défini dans la démonstration du Lemme 2.2.2, contiendrait un motif interdit pour le décalage \mathbf{T} . En particulier, la configuration x contiendrait aussi un motif interdit pour le décalage \mathbf{T} , c'est-à-dire $x \notin \mathbf{T}$ ce qui est contradictoire. Finalement la configuration y ne contient aucun motif de l'ensemble F' , et donc $y \in \mathbf{T}_{F'}$.

- $\mathbf{T}_{F'} \subseteq \mathbf{SA}_{\mathbb{G}}(\mathbf{T})$

Soit $y \in \mathbf{T}_{F'}$. Pour tout entier $n \in \mathbb{N}$ on a $y_{\mathbb{S}_n^{d'}} \notin F'$, donc il existe un motif $p_n \in \Phi(y_{\mathbb{S}_n^{d'}}) \cap A_n^{\mathbb{S}_n^{d'}}$ qui ne contient aucun motif interdit pour le décalage \mathbf{T} . Par compacité, on peut construire un élément $x \in \mathbf{T}$ tel que $y_i = x_{i_1 u_1 + \dots + i_{d'} u_{d'}}$ pour tout $i = (i_1, \dots, i_{d'}) \in \mathbb{Z}^{d'}$. □

L'ensemble récursivement énumérable F' est tel que $\mathbf{T}_{F'} = \mathbf{SA}_{\mathbb{G}}(\mathbf{T})$, donc les décalages effectifs sont stables par sous-action projective. □

Dans [PS10], les auteurs étudient la classe $\mathcal{Cl}_{\mathbf{SA}}(\mathcal{SFT})$ et montrent qu'elle contient tous les décalages sofiques d'entropie strictement positive. Cependant la classe n'est pas complètement caractérisée, et savoir quels sont les décalages qui peuvent être obtenus par sous-action d'un décalage de type fini reste une question ouverte (l'exemple 2.2.3 montre que $\mathcal{Cl}_{\mathbf{SA}}(\mathcal{SFT})$ contient des décalages non sofiques).

2.3 Décalage sofique codant une machine de Turing

Cette partie présente une construction d'un décalage sofique sur \mathbb{Z}^2 dans lequel apparaît le diagramme espace-temps d'une machine de Turing à ruban semi-infini. Cette construction est celle présentée dans [AS10].

2.3.1 Machines de Turing dans un SFT

Nous commençons par montrer en quoi les décalages de type fini constituent un bon moyen de visualiser l'évolution des machines de Turing, à l'aide des diagrammes espace-temps de ces dernières.

On reprend ici la Définition 8 des machines de Turing dans le cas particulier où le ruban de la machine est le graphe de Cayley de \mathbb{N} .

Définition 13. Une machine de Turing est la donnée de $\mathcal{M} = (Q, \Gamma, \#, q_0, \delta, Q_F)$ où :

- Q est un l'ensemble fini des états possibles pour la tête de calcul ; $q_0 \in Q$ est l'état initial ;
- Γ est un alphabet fini ;
- $\# \notin \Gamma$ est le symbole blanc ;

- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, \cdot, \rightarrow\}$ est la fonction de transition. Étant donné un état de la tête de calcul et la lettre inscrite sur le ruban à la position de la tête de calcul, cette lettre est remplacée par une nouvelle lettre, la tête de calcul se déplace sur une cellule adjacente (ou bien reste en place) et passe dans un nouvel état ;
- $F \subset Q_F$ est l'ensemble des états finaux. Lorsqu'un état final est atteint, la machine arrête son calcul.

Exemple 2.3.1. Considérons la machine de Turing \mathcal{M}_{ex} qui énumère sur son ruban les mots $ab, aabb, aaabbb, \dots, a^n b^n, \dots$ et ne s'arrête jamais. Cette machine travaille sur l'alphabet à trois lettres $\Gamma = \{a, b, \parallel\}$ et la tête de calcul peut être dans cinq états $Q = \{q_0, q_{a+}, q_{b+}, q_{b++}, q_{\parallel}\}$. Lorsque la tête de calcul est dans l'état q_{a+} , elle va ajouter un symbole a au mot déjà inscrit sur le ruban. Dans l'état q_{b++} , la machine va ajouter deux b consécutifs, et pour cela elle passera par l'état intermédiaire q_{b+} . Enfin l'état q_{\parallel} est atteint chaque fois qu'un mot de la forme $a^n b^n$ est inscrit sur le ruban. Un symbole de séparation \parallel est inscrit sur le ruban à la fin de chaque mot de la forme $a^n b^n$. La fonction de transition δ_{ex} est donnée par les règles suivantes :

$$\left. \begin{array}{l} \delta_{\text{ex}}(q_0, \#) = (q_{b+}, a, \rightarrow) \\ \delta_{\text{ex}}(q_{b+}, \#) = (q_{\parallel}, b, \rightarrow) \\ \delta_{\text{ex}}(q_{\parallel}, \#) = (q_{\parallel}, \parallel, \cdot) \end{array} \right\} \begin{array}{l} \text{Initialisation du ruban : la machine écrit le premier} \\ \text{mot } ab \text{ sur le ruban et positionne sa tête de calcul sur} \\ \text{le symbole de séparation } \parallel \text{ à la droite du mot } ab. \end{array}$$

$$\left. \begin{array}{l} \delta_{\text{ex}}(q_{\parallel}, \parallel) = (q_{\parallel}, \parallel, \leftarrow) \\ \delta_{\text{ex}}(q_{\parallel}, b) = (q_{\parallel}, b, \leftarrow) \\ \delta_{\text{ex}}(q_{\parallel}, a) = (q_{a+}, a, \rightarrow) \end{array} \right\} \begin{array}{l} \text{Si un mot } a^n b^n \parallel \text{ est écrit sur le ruban, et que la tête de} \\ \text{calcul se trouve sur le symbole } \parallel \text{ et est dans l'état } q_{\parallel}, \\ \text{alors la machine recherche la lettre } a \text{ la plus à droite} \\ \text{dans le mot } a^n b^n. \end{array}$$

$$\left. \begin{array}{l} \delta_{\text{ex}}(q_{a+}, b) = (q_{b++}, a, \rightarrow) \\ \delta_{\text{ex}}(q_{b++}, b) = (q_{b++}, b, \rightarrow) \\ \delta_{\text{ex}}(q_{b++}, \parallel) = (q_{b+}, b, \rightarrow) \end{array} \right\} \begin{array}{l} \text{La machine remplace le lettre } b \text{ la plus à gauche par} \\ \text{une lettre } a \text{ puis cherche le symbole de séparation } \parallel \\ \text{à droite du mot. Une fois ce symbole atteint, la ma-} \\ \text{chine le remplace par le mot } bb \parallel, \text{ de sorte que le mot} \\ a^{n+1} b^{n+1} \parallel \text{ est à présent écrit sur le ruban, puis la} \\ \text{tête de calcul se place sur le symbole } \parallel \text{ et passe dans} \\ \text{l'état } q_{\parallel}. \end{array}$$

Un calcul de cette machine (qui débute toujours avec le mot vide sur le ruban, comme précisé plus haut) passera nécessairement par les configurations du ruban de la Figure 10 (où le temps s'écoule de bas en haut).

...
...	#	a	a	a	a	b	$(q_{b^{++}}, b)$		#	#	...
...	#	a	a	a	a	$(q_{b^{++}}, b)$	b		#	#	...
...	#	a	a	a	(q_{a^+}, b)	b	b		#	#	...
...	#	a	a	$(q_{ }, a)$	b	b	b		#	#	...
...	#	a	a	a	$(q_{ }, b)$	b	b		#	#	...
...	#	a	a	a	b	$(q_{ }, b)$	b		#	#	...
...	#	a	a	a	b	b	$(q_{ }, b)$		#	#	...
...	#	a	a	a	b	b	b	$(q_{ },)$	#	#	...
...	#	a	a	a	b	b	b	$(q_{ }, \#)$	#	#	...
...	#	a	a	a	b	b	$(q_{b^+}, \#)$	#	#	#	...
...	#	a	a	a	b	$(q_{b^{++}},)$	#	#	#	#	...
...	#	a	a	a	$(q_{b^{++}}, b)$		#	#	#	#	...
...	#	a	a	(q_{a^+}, b)	b		#	#	#	#	...
...	#	a	$(q_{ }, a)$	b	b		#	#	#	#	...
...	#	a	a	$(q_{ }, b)$	b		#	#	#	#	...
...	#	a	a	b	$(q_{ }, b)$		#	#	#	#	...
...	#	a	a	b	b	$(q_{ },)$	#	#	#	#	...
...	#	a	a	b	b	$(q_{ }, \#)$	#	#	#	#	...
...	#	a	a	b	$(q_{b^+}, \#)$	#	#	#	#	#	...
...	#	a	a	$(q_{b^{++}},)$	#	#	#	#	#	#	...
...	#	a	(q_{a^+}, b)		#	#	#	#	#	#	...
...	#	$(q_{ }, a)$	b		#	#	#	#	#	#	...
...	#	a	$(q_{ }, b)$		#	#	#	#	#	#	...
...	#	a	b	$(q_{ },)$	#	#	#	#	#	#	...
...	#	a	b	$(q_{ }, \#)$	#	#	#	#	#	#	...
...	#	a	$(q_{b^+}, \#)$	#	#	#	#	#	#	#	...
...	#	$(q_0, \#)$	#	#	#	#	#	#	#	#	...

FIGURE 10: Exemple de calcul d'une machine de Turing qui énumère sur son ruban tous les mots de la forme $a^n b^n$ pour tout $n \in \mathbb{N}$.

Le comportement d'une machine de Turing, c'est-à-dire son ensemble de règles, est codé dans un décalage de type fini dont on donne les motifs autorisés. La dimension horizontale joue le rôle du ruban de la machine, tandis que la dimension verticale représente l'évolution du temps (qui s'écoule de bas en haut). On obtient ainsi les *diagrammes espace-temps* de la machine \mathcal{M} , que l'on peut construire à l'aide des motifs autorisés de taille 3×2 suivants :

- si le motif code une partie du ruban dans laquelle la tête de calcul n'apparaît pas, les deux lignes du motif sont identiques et pour $x, y, z \in \Gamma$ on autorise le motif :

x	y	z
x	y	z

- si la tête de calcul est présente dans la partie du ruban, on code les règles de transition de la machine. Par exemple la règle $\delta(q_1, x) = (q_2, y, \leftarrow)$ sera codée par les motifs :

v	w	(q_2, z)
v	w	z

w	(q_2, z)	y
w	z	(q_1, x)

(q_2, z)	y	z'
z	(q_1, x)	z'

y	z'	z''
(q_1, x)	z'	z''

- enfin si q_f est un état final, alors la machine arrête son calcul ce que l'on traduit par le motif :

z	(q_f, x)	z'
z	(q_f, x)	z'

On note par $P_{\mathcal{M}}$ l'ensemble des motifs interdits sur l'alphabet $A_{\mathcal{M}} = \Gamma \cup (Q \times \Gamma)$ construit à partir des règles de transition de \mathcal{M} , c'est-à-dire les motifs qui ne sont pas représentés dans la liste ci-dessus.

Considérons à présent le décalage de type fini $\mathbf{T}_{P_{\mathcal{M}}}$. Il contient tous les diagrammes espace-temps de la machine \mathcal{M} , mais aussi d'autres configurations qui ne sont jamais atteintes par la machine. Avec la machine de Turing de l'exemple 2.3.1, le SFT $\mathbf{T}_{P_{\mathcal{M}_{\text{ex}}}}$ contient une configuration dans laquelle le ruban est dans l'état suivant

...	#	...	#	a	b	b	b	b	b	$(q_{\parallel}, \parallel)$	#	...	#	...
-----	---	-----	---	-----	-----	-----	-----	-----	-----	------------------------------	---	-----	---	-----

état du ruban qui est incohérent puisque jamais atteint dans un calcul de \mathcal{M}_{ex} .

Le problème vient du manque d'information sur le début d'un calcul. Il faut spécifier un point dans \mathbb{Z}^2 qui jouera le rôle d'origine du calcul : à cette position la tête de calcul est dans l'état initial q_0 , placée au début du mot d'entrée, en dehors duquel le ruban ne contient que des symboles blanc #. Ainsi on s'assure que la configuration du décalage $\mathbf{T}_{P_{\mathcal{M}}}$ produite avec cette contrainte correspond bien à un diagramme espace-temps de la machine \mathcal{M} .

Comment introduire une origine dans un décalage? Par compacité du décalage, il est impossible d'imposer qu'un symbole origine apparaisse une fois et une seule dans chaque configuration du décalage. Dans la Partie 2.3.2 nous allons voir comment résoudre ce problème, en construisant des espaces de calculs finis (et dont on peut donc facilement pointer l'origine) de sorte que chaque configuration contient des espaces de calcul de taille arbitrairement grande, permettant ainsi de voir apparaître les diagrammes espace-temps de la machine. Pour cela on utilise des décalages substitutifs, définis dans la Partie 2.1.

2.3.2 Espaces de calcul pour MT

Nous allons présenter une construction d'espaces de calcul pour des machines de Turing. Rappelons que les machines considérées travaillent sur un ruban semi-infini (la tête de lecture ne peut pas se déplacer à gauche de sa position initiale). Dans cette partie, un espace de calcul pour une machine de Turing \mathcal{M} est un rectangle dans le plan discret. On cherche à construire un décalage $\mathbf{T}_{\text{Calcul}}$ avec les propriétés suivantes :

- $\mathbf{T}_{\text{Calcul}}$ est sofique ;
- toute configuration de $\mathbf{T}_{\text{Calcul}}$ contient des espaces de calcul de taille arbitrairement grande, en espace et en temps.

Un tel décalage assure de voir apparaître les diagrammes espace-temps de la machine \mathcal{M} .

On choisit ici d'utiliser un décalage substitutif pour définir les espaces de calcul. Soit l'alphabet \mathcal{G}_1 à quatre éléments présenté dans la Figure 11. Cet alphabet $\mathcal{G}_1 = \{ \square, \blacksquare, \blacktriangleright, \blacktriangleleft \}$ peut être divisé en deux parties. En référence aux tuiles de Wang, les lettres de cet alphabet \mathcal{G}_1 seront aussi appelées tuiles. Les lettres \blacksquare , \blacktriangleright et \blacktriangleleft correspondent aux *tuiles de calcul*, où les calculs de machine de Turing seront effectivement implémentés, tandis que la lettre \square correspond aux *tuiles de communication* par lesquelles de l'information peut circuler. Plus précisément \blacktriangleright et \blacktriangleleft sont des *tuiles de calcul de bord*. Sur cet alphabet \mathcal{G}_1 on définit une substitution s_1 par les règles données dans la Figure 11.

On dit qu'une substitution s est à *dérivation unique* si pour tout $x \in \mathbf{T}_{\{s\}}$, il existe un unique $y \in \mathbf{T}_{\{s\}}$ tel que $x = s(y)$, c'est-à-dire que la substitution s est injective sur le décalage $\mathbf{T}_{\{s\}}$.

Proposition 2.3.1. *La substitution s_1 est à dérivation unique.*

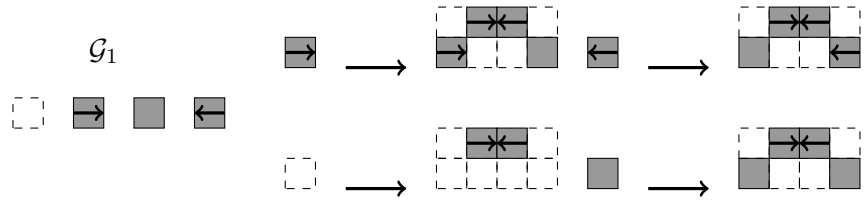


FIGURE 11: L'alphabet \mathcal{G}_1 utilisé pour définir des zones de calcul, qui comprend des tuiles de calcul et des tuiles de communication, ainsi que les règles de la substitution s_1 .

Démonstration. Le motif $\begin{smallmatrix} \rightarrow & * \\ \leftarrow & \end{smallmatrix}$ apparaît dans chacune des règles de la substitution s_1 . Pour toute configuration $x \in \mathbf{T}_{\{s_1\}}$, il existe donc $(i, j) \in [0, 3] \times [0, 1]$ tel que $x_{\{n_1+i+1\} \times [n_2+j+1, n_2+j+2]} = \begin{smallmatrix} \rightarrow & * \\ \leftarrow & \end{smallmatrix}$ pour tout $(n_1, n_2) \in \mathbb{N} \times \mathbb{N}$. De plus, ce motif ne peut pas apparaître à une autre position, donc le couple d'entier (i, j) est choisi de manière unique. Considérons la partition du plan $([n_1+i, n_1+i+3] \times [n_2+j, n_2+j+1])_{(n_1, n_2) \in 4\mathbb{N} \times 2\mathbb{N}}$ pour la configuration x . Comme toutes les tuiles ont des images différentes par la substitution s_1 , cette partition donne un unique antécédent y à x par s_1 . On en déduit que $s_1(y) = \sigma^{(i,j)}(x)$; la substitution s_1 est donc bien à dérivation unique. \square

Cette substitution s_1 définit un décalage $\mathbf{T}_{\{s_1\}}$, sofique en vertu du Théorème 2.1.1. On décrit à présent comment des zones de calcul apparaissent dans les configurations de ce décalage. Dans cette construction, l'espace est représenté par la coordonnée horizontale, tandis que le temps est représenté par la coordonnée verticale, et s'écoule du bas vers le haut. Observons tout d'abord les motifs obtenus en itérant s_1 sur une lettre (voir la Figure 12).

Sur une ligne horizontale, une *zone de calcul* est constituée d'un groupe de tuiles de calcul \blacksquare situées entre une tuile \rightarrow à gauche et une tuile \leftarrow à droite. La *taille* de la zone de calcul est le nombre de tuiles de calcul (\rightarrow , \blacksquare ou \leftarrow) qui constituent la zone.

Considérons une configuration $x \in \mathbf{T}_{\{s_1\}}$ ainsi qu'une zone de calcul dans x . Comme la substitution s_1 est à dérivation unique (voir la Proposition 2.3.1), pour tout entier n il existe une manière unique de partitionner la configuration x en rectangles de taille $4^n \times 2^n$, de sorte que chacun de ces rectangles est de la forme $s_1^n(a)$ pour une certaine lettre $a \in \mathcal{G}_1$. Il existe donc un plus petit entier n tel que la zone de calcul de x apparaît dans $s_1^n(a)$ pour une lettre $a \in \mathcal{G}_1$. Cet

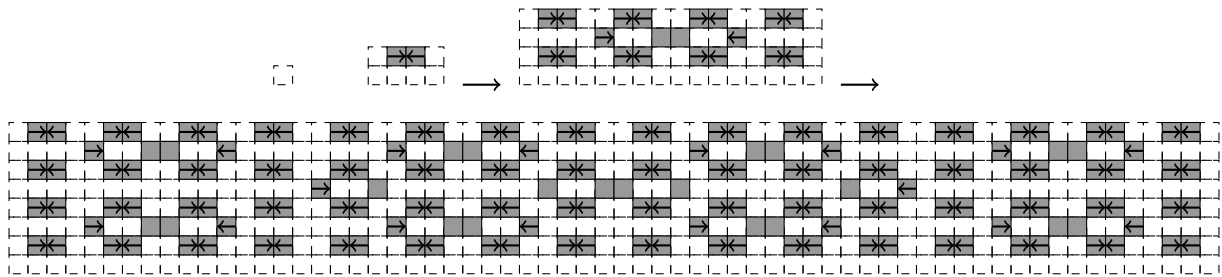


FIGURE 12: Trois itérations de la substitution s_1 en partant de la lettre $[]$.

entier minimal sera appelé le *niveau* de la zone de calcul.

En itérant la substitution s_1 sur un motif de \mathcal{G}_1 , on obtient des zones de calculs aussi larges qu'on le souhaite (voir la Figure 13 pour un exemple).

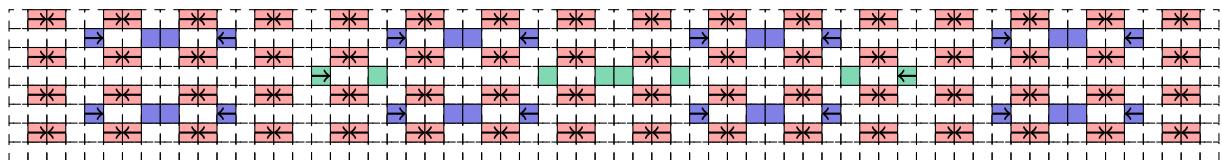


FIGURE 13: Zones de calcul de différents niveaux (les zones de niveau 1 en rouge, celles de niveau 2 en bleu et celle de niveau 3 en vert) après trois itérations de s_1 .

Après n itérations de la substitution s_1 sur la lettre $[]$ (on n'importe quelle autre lettre de l'alphabet \mathcal{G}_1 , la description étant la même à part la ligne inférieure qui change), ce qui revient à considérer le motif $s^n([])$, on obtient un rectangle de taille $4^n \times 2^n$. Par récurrence, on peut montrer que pour tout $m \in [1, n]$, on obtient $4^{n-m} * 2^{n-m} = 8^{n-m}$ zones de calcul de niveau m dans le motif $s^n([])$, la taille de ces zones de calcul étant 2^m . Plus précisément, si sur la ligne $j \in [0, 2^n - 1]$ de $s^n([])$ se trouve une zone de calcul de niveau m , alors sur cette ligne il y a 4^{n-m} zones de calcul de niveau m . De plus pour chaque tuile de calcul située aux coordonnées (i, j) , la prochain tuile de calcul de même niveau située dans la même colonne est séparée de celle-ci par $2^m - 1$ tuiles de

communication et se trouve donc aux coordonnées $(i, j + 2^m)$; cette tuile de calcul de niveau m est dans la même position relative à sa zone de calcul que la tuile en position (i, j) . Le même phénomène se produit si l'on regarde vers le bas. L'ensemble des zones de calcul de la même taille 2^m alignées verticalement forment une *bande de calcul* de taille 2^m (la taille est en fait la largeur de la bande de calcul).

Remarque. Il est possible que sur une ligne, un symbole \Rightarrow apparaisse sans symbole \Leftarrow à sa droite. Cette zone de calcul dégénérée sera appelée zone de calcul infinie, et on considère qu'elle est de niveau $+\infty$.

Proposition 2.3.2. *Considérons une configuration $x \in \mathbf{T}_{\{s_1\}}$ et C_n une zone de calcul de niveau n dans x . Supposons aussi que C_n apparaît dans la $i^{\text{ième}}$ ligne de x , que l'on notera $x_{\mathbb{Z} \times \{i\}}$. Alors C_n vérifie les propriétés suivantes :*

1. la zone de calcul C_n contient 2^n tuiles de calcul, séparées par des tuiles de communication;
2. sur la ligne $x_{\mathbb{Z} \times \{i\}}$ il n'y a que des zones de calcul de niveau n , séparées les unes des autres par 2^{2n-1} tuiles de communication;
3. la ligne $x_{\mathbb{Z} \times \{i\}}$ est répétée verticalement toutes les 2^n lignes, autrement dit $x_{\mathbb{Z} \times \{i\}} = x_{\mathbb{Z} \times \{i+k \times 2^n\}}$ pour tout entier $k \in \mathbb{Z}$;
4. selon la coordonnée verticale, entre deux tuiles de calcul consécutives des lignes $x_{\mathbb{Z} \times \{i\}}$ et $x_{\mathbb{Z} \times \{i+k \times 2^n\}}$, on trouve uniquement des tuiles de communication, au nombre de $2^n - 1$.

Voyons à présent comment deux tuiles de calcul d'une même bande de calcul peuvent communiquer.

2.3.3 Communication dans le décalage $\mathbf{T}_{\{s_1\}}$

On décrit dans cette partie comment les tuiles de communication sont utilisées.

Principe de communication dans un décalage Un *canal de communication* est une suite de tuiles de communication adjacentes. Ce canal commence et se termine par une tuile de calcul.

Un *transfert d'information* est formé de trois données :

- un canal de communication c ;
- une *tuile de calcul initiale* appelée i et une *tuile de calcul finale* appelée f :

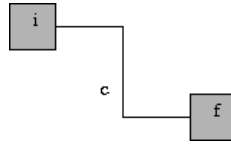


FIGURE 14: Un canal de communication entre les tuiles de calcul i et f .

- des règles locale qui déterminent le symbole qui sera transféré à travers le canal de communication, qui dépend à la fois de la direction du canal partant de i (resp. arrivant sur f) et du symbole contenu par la tuile i (resp. f).

Deux tuiles de communication voisines portent le même symbole, qui est transféré le long du canal de communication. Précisons que les tuiles i et f ne contiennent pas forcément la même lettre (par exemple la règle locale peut modifier le symbole transféré à son arrivée sur la tuile f). Une même tuile de calcul peut se trouver aux extrémités de canaux de communication différents, mais on impose ici que ce nombre de canaux soit borné.

Étant donné un décalage \mathbf{T} contenant des canaux de communication, il est possible de coder des transferts d'information dans un décalage \mathbf{T}_{info} . Si les symboles transférés dépendent uniquement d'un motif fini autour des tuiles initiales et finales de chaque canal, et que \mathbf{T} est un décalage de type fini (resp. sofique), alors \mathbf{T}_{info} est aussi de type fini (resp. sofique).

Communication à l'intérieur d'une bande de calcul Deux tuiles de calcul dans une même bande de niveau n situées sur une même colonne peuvent communiquer grâce aux tuiles de communication des $2^n - 1$ lignes intermédiaires (transfert vertical de l'information). Au sein d'une même zone de calcul de niveau n , deux tuiles de calcul adjacentes (c'est-à-dire qu'entre elles deux il n'y a que des tuiles de communication) peuvent communiquer via les $2 * 4^{n-1} - 2^n$ tuiles de communication qui les séparent (transfert horizontal de l'information). La Figure 15 illustre ces propos.

Ainsi chaque tuile de communication contient une partie de deux canaux, l'un horizontal pour permettre la communication au sein d'une même zone de calcul, l'autre vertical pour permettre la communication au sein d'une même bande de calcul.

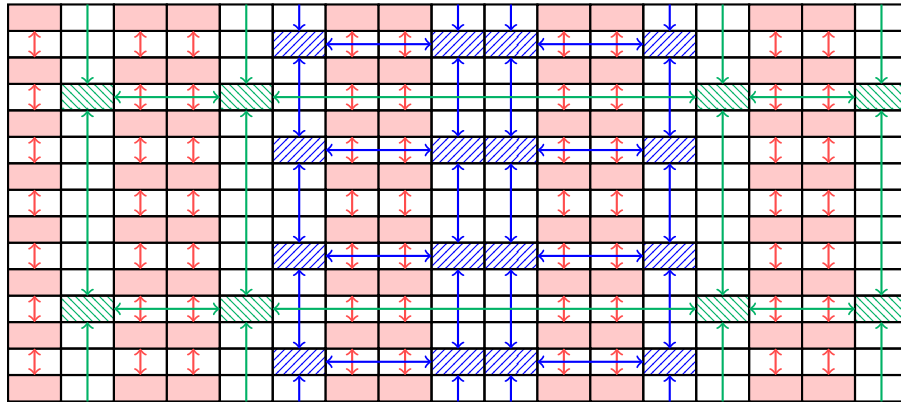


FIGURE 15: Les zones de calcul de niveaux 1,2 et 3 sont signalées par des couleurs différentes. Les communications entre tuiles de calcul de ces différents niveaux sont représentées par des flèches.

2.3.4 Initialisation des calculs

Les bandes de calcul décrites dans la partie précédente sont restreintes en espace mais pas en temps. De ce fait des configurations inconsistantes de la machine de Turing peuvent apparaître. Pour résoudre ce problème, on munit chacune des bandes de calcul d'une horloge qui sera réinitialisée de façon périodique. A chaque étape de calcul, l'horloge est incrémentée et lorsqu'elle est réinitialisée, la machine de Turing recommence un nouveau calcul.

On utilise l'alphabet à quatre éléments $\mathcal{C} = \{0, 1, \emptyset, \sim\}$ pour construire un décalage sofique $\mathbf{T}_{\text{Horloge}}$ obtenu en ajoutant des règles de type fini au décalage sofique $\mathbf{Prod}(\mathbf{T}_{s_1}, \mathcal{C}^{\mathbb{Z}^2})$, où $\mathbf{T}_{\{s_1\}}$ est le décalage sofique décrit dans la Partie 2.3.2.

On note par $\pi_{\mathcal{C}}$ la projection sur la seconde coordonnée. L'horloge est en fait un automate fini qui simule l'addition binaire modulo 2^{2^n} sur un ruban de 2^n tuiles. Le symbole spécial \emptyset correspond à la retenue dans l'addition binaire, et le symbole \sim est utilisé pour synchroniser des zones de calculs adjacentes de même niveau. Pour empêcher l'apparition d'états inconsistants dans l'horloge, on interdit les motifs $\begin{bmatrix} \emptyset & 0 \\ \emptyset & 1 \end{bmatrix}$, $\begin{bmatrix} \emptyset & 1 \\ 0 & \emptyset \end{bmatrix}$, $\begin{bmatrix} x & \sim \\ \sim & x \end{bmatrix}$ et $\begin{bmatrix} \sim & x \\ x & \sim \end{bmatrix}$ où $x \in \{0, 1, \emptyset\}$; on notera par **Consist** cette condition de type fini.

Les autres conditions de type fini **Compt** sur l'alphabet $\mathcal{G}_1 \times \mathcal{C}$, qui servent à décrire le processus d'addition, sont décrites dans la Figure 16.

Les horloges des différents niveaux de calcul évoluent selon les règles dé-

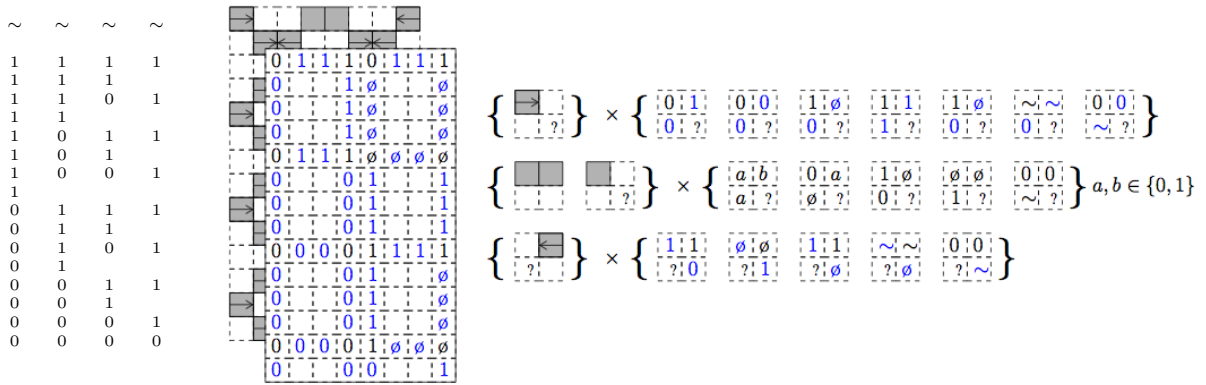


FIGURE 16: À gauche, un exemple d'évolution de l'horloge sur une zone de calcul de niveau 2. Au centre l'évolution de cette horloge sur une bande de calcul de niveau 2 : les temps 001∅, 0011, 01∅∅ et 0101 sont écrits successivement. À droite certaines des règles locales pour coder l'horloge, données par les motifs autorisés.

crites dans la Figure 16, et lorsqu'un symbole \emptyset atteint la tuile de calcul la plus à gauche \Rightarrow , les horloges sont réinitialisées. Avant la réinitialisation proprement dite (retour dans la configuration dans laquelle le ruban ne contient que des symboles 0), l'horloge passe par une configuration dans laquelle son ruban ne contient que des symboles \sim . Par exemple une horloge associée à une bande de calcul de niveau 1 passera par les configurations successives 00, 01, 1∅, 11, ∅∅, $\sim\sim$, 00, ... Ainsi une horloge associée à une zone de calcul de niveau n est réinitialisée après $2^{2^n} + 2$ étapes de calcul.

Grâce à la configuration ne contenant que des symboles \sim , il est possible de synchroniser une horloge sur une bande de calcul de niveau n avec les horloges de ses deux bandes de calcul voisines de niveau n , et ce uniquement par des règles locales que l'on notera par **Synchro**. Plus précisément, ces règles locales imposent qu'une horloge soit dans la configuration $\sim \dots \sim$ uniquement lorsque ses deux voisines sont dans la même configuration (un signal portant le symbole \sim étant envoyé par le canal de communication reliant deux bandes de calcul voisines).

Les règles de type fini **Synchro** assurent que sur une même ligne les horloges qui apparaissent, et qui correspondent donc à des bandes de calcul de même niveau, sont synchronisées (sur une même ligne, toutes les horloges sont dans la même configuration). On obtient ainsi un décalage sofique $\mathbf{T}_{\text{Horloge}}$ dans lequel chaque bande de calcul du décalage $\mathbf{T}_{\{s_1\}}$ est à présent munie d'une horloge.

Notons qu'en aucune manière on n'impose à des horloges de bandes de calcul de niveaux différents d'être synchronisées d'aucune façon que ce soit.

$$\mathbf{T}_{\text{Horloge}} = \mathbf{TF}_{\text{Compt} \cup \text{Consist} \cup \text{Synchro}} \left(\mathbf{Prod} \left(\mathbf{T}_{\{s_1\}}, \mathcal{C}^{\mathbb{Z}^2} \right) \right)$$

Proposition 2.3.3. *Le décalage sofique $\mathbf{T}_{\text{Horloge}}$ est tel que, à l'intérieur d'une bande de calcul de niveau n de largeur 2^n , une horloge est réinitialisée toutes les $2^{2^n} + 2$ étapes de calcul.*

2.3.5 Diagramme espace-temps d'une MT dans un décalage sofique

Le décalage sofique $\mathbf{T}_{\text{Horloge}}$ présenté dans la Partie 2.3.4 est ici utilisé pour construire un décalage sofique dans lequel apparaissent les diagrammes espace-temps d'une machine de Turing \mathcal{M} . Pour cela les règles $P_{\mathcal{M}}$, définies dans la Partie 2.3.1, sont ajoutées au décalage $\mathbf{T}_{\text{Horloge}}$. La difficulté est que les espaces de calcul présents dans les configurations de ce décalage sont fractionnés, et que l'on ne peut donc pas appliquer directement les règles locales $P_{\mathcal{M}}$. Il faut alors les modifier au préalable pour tenir compte des canaux de communication décrits dans la Partie 2.3.2.

Partant du décalage sofique $\mathbf{T}_{\text{Horloge}}$ défini dans la Partie 2.3.4, on l'insère dans le décalage $\mathbf{Prod} \left(\mathbf{T}_{\text{Horloge}}, \tilde{A}^{\mathbb{Z}^2} \right)$ où le nouvel alphabet \tilde{A} est défini de la manière suivante : $\tilde{A} = A_{\mathcal{M}} \cup (A_{\mathcal{M}} \times A_{\mathcal{M}} \times A_{\mathcal{M}})$. Un symbole de \tilde{A} représente donc soit un symbole de $A_{\mathcal{M}}$ sur une tuile de calcul, soit la superposition de trois symboles de $A_{\mathcal{M}}$ transférés (horizontalement pour le premier et le deuxième, verticalement pour le troisième) par une tuile de communication. On définit $\pi_{\mathcal{G}_1}$ et $\pi_{\mathcal{C}}$ les projections respectivement sur \mathcal{G}_1 et \mathcal{C} pour la première coordonnée de $\mathbf{Prod} \left(\mathbf{T}_{\text{Horloge}}, \tilde{A}^{\mathbb{Z}^2} \right)$. De plus on appellera $\pi_{\tilde{A}}$ la projection sur la deuxième coordonnée de $\mathbf{Prod} \left(\mathbf{T}_{\text{Horloge}}, \tilde{A}^{\mathbb{Z}^2} \right)$. Pour une tuile de communication, on peut ainsi écrire $\pi_{\tilde{A}_1}$, $\pi_{\tilde{A}_2}$ et $\pi_{\tilde{A}_3}$ pour respectivement la première, deuxième et troisième coordonnées de $A_{\mathcal{M}} \times A_{\mathcal{M}} \times A_{\mathcal{M}}$.

On ajoute des conditions de type fini au décalage sofique $\mathbf{Prod} \left(\mathbf{T}_{\text{Horloge}}, \tilde{A}^{\mathbb{Z}^2} \right)$. Ces conditions s'expriment par des motifs dont le support est de la forme

$$\begin{array}{|c|c|c|} \hline & a & \\ \hline b & c & d \\ \hline & e & \\ \hline \end{array} \quad \text{avec } a, b, c, d, e \in \mathcal{G}_1 \times \mathcal{C} \times \tilde{A}$$

Les conditions sont les suivantes :

- si la tuile centrale correspond à une tuile de communication dans $\mathbf{T}_{\text{Horloge}}$, autrement dit $\pi_{\mathcal{G}_1}(c) = \square$, on applique l'ensemble de conditions **Transfert** : les première et deuxième coordonnées sont constantes le long de la ligne centrale, tandis que la troisième coordonnée est constante le long de la colonne centrale. Plus précisément, $\pi_{\tilde{A}_1}(b) = \pi_{\tilde{A}_1}(c) = \pi_{\tilde{A}_1}(d)$, $\pi_{\tilde{A}_2}(b) = \pi_{\tilde{A}_2}(c) = \pi_{\tilde{A}_2}(d)$ et $\pi_{\tilde{A}_3}(a) = \pi_{\tilde{A}_3}(c) = \pi_{\tilde{A}_3}(e)$. Ces conditions sont valables si toutes les tuiles adjacentes sont des tuiles de communication. Si une tuile de calcul est présente dans le voisinage, on utilise la projection $\pi_{\tilde{A}}$;
- si la tuile centrale correspond à une tuile de calcul dans $\mathbf{T}_{\text{Horloge}}$, autrement dit $\pi_{\mathcal{G}_1}(c) \in \{ \blacksquare, \blacktriangleright, \blacktriangleleft \}$, on utilise l'une des conditions suivantes selon le cas :
 - conditions **Init** : lorsque l'horloge est dans son état initial $0 \dots 0$, chaque tuile contient un symbole blanc \sharp et la tête de calcul est dans l'état initial q_0 , positionnée sur la tuile de calcul \blacktriangleright , ce que l'on exprime par :
 - * si $\pi_{\mathcal{C}}(c) = \sim$ et $\pi_{\mathcal{G}_1}(c) = \blacktriangleright$ alors $\pi_{\tilde{A}}(c) = \pi_{\tilde{A}_1}(d) = \pi_{\tilde{A}_2}(b) = \pi_{\tilde{A}_3}(a) = (q_0, \sharp)$,
 - * si $\pi_{\mathcal{C}}(c) = \sim$ et $\pi_{\mathcal{G}_1}(c) \in \{ \blacksquare, \blacktriangleleft \}$ alors $\pi_{\tilde{A}}(c) = \pi_{\tilde{A}_1}(d) = \pi_{\tilde{A}_2}(b) = \pi_{\tilde{A}_3}(a) = \sharp$;
 - conditions **Comp** : si l'horloge n'est pas dans son état initial, on utilise les règles décrites dans $\mathcal{P}_{\mathcal{M}}$. Plus précisément :
 - * si $\pi_{\mathcal{C}}(c) \neq \sim$ et $\pi_{\mathcal{G}_1}(c) = \blacksquare$ alors

$$\begin{array}{|c|c|c|} \hline & \pi_{\tilde{A}_3}(a) & \\ \hline \pi_{\tilde{A}_1}(b) & \pi_{\tilde{A}}(c) & \pi_{\tilde{A}_2}(d) \\ \hline \end{array} \in P_{\mathcal{M}}, \quad \pi_{\tilde{A}}(c) = \pi_{\tilde{A}_2}(b) = \pi_{\tilde{A}_1}(d) \text{ et } \pi_{\tilde{A}}(c) = \pi_{\tilde{A}_3}(e),$$

- * si $\pi_{\mathcal{C}}(c) \neq \sim$, $\pi_{\mathcal{G}_1}(c) = \blacktriangleright$ et si la troisième coordonnées de $\delta(\pi_{\tilde{A}}(c))$ n'est pas \leftarrow , c'est-à-dire que la fonction de transition de la machine de Turing ne déplace pas la tête de calcul vers la gauche, alors

$$\begin{array}{|c|c|c|} \hline & \pi_{\tilde{A}_3}(a) & \\ \hline \sharp & \pi_{\tilde{A}}(c) & \pi_{\tilde{A}_2}(d) \\ \hline \end{array} \in P_{\mathcal{M}}, \quad \pi_{\tilde{A}}(c) = \pi_{\tilde{A}_2}(b) = \pi_{\tilde{A}_1}(d) \text{ and } \pi_{\tilde{A}}(c) = \pi_{\tilde{A}_3}(e),$$

- * si $\pi_{\mathcal{C}}(c) \neq \sim$, $\pi_{\mathcal{G}_1}(c) = \blacktriangleleft$ et si la troisième coordonnée de $\delta(\pi_{\tilde{A}}(c))$ n'est pas \rightarrow , c'est-à-dire que la fonction de transition

de la machine de Turing ne déplace pas la tête de calcul vers la droite, alors

$$\begin{array}{|c|c|c|} \hline & \pi_{\tilde{A}_3}(a) & \\ \hline \pi_{\tilde{A}_1}(b) & \pi_{\tilde{A}}(c) & \# \\ \hline \end{array} \in P_{\mathcal{M}}, \quad \pi_{\tilde{A}}(c) = \pi_{\tilde{A}_2}(b) = \pi_{\tilde{A}_1}(d) \text{ et } \pi_{\tilde{A}}(c) = \pi_{\tilde{A}_3}(e);$$

– conditions **Bord** : si la tête de calcul veut se déplacer à gauche d'une tuile de calcul \blacktriangleright ou à droite d'une tuile de calcul \blacktriangleleft , elle entre dans un état spécial q_{wait} et le calcul est suspendu jusqu'à la prochaine réinitialisation de l'horloge. Plus précisément :

* si $\pi_{\mathcal{C}}(c) \neq \sim$, $\pi_{\mathcal{G}_1}(c) = \blacktriangleright$ et si la troisième coordonnée de $\delta(\pi_{\tilde{A}}(c))$ est \leftarrow , alors

$$\begin{array}{|c|c|} \hline q_{\text{wait}} & \\ \hline \pi_{\tilde{A}}(c) & \pi_{\tilde{A}_2}(d) \\ \hline \end{array}, \quad \pi_{\tilde{A}}(c) = \pi_{\tilde{A}_2}(b) = \pi_{\tilde{A}_1}(d) \text{ and } \pi_{\tilde{A}}(c) = \pi_{\tilde{A}_3}(e);$$

* si $\pi_{\mathcal{C}}(c) \neq \sim$, $\pi_{\mathcal{G}_1}(c) = \blacktriangleleft$ et si la troisième coordonnée de $\delta(\pi_{\tilde{A}}(c))$ est \rightarrow , alors

$$\begin{array}{|c|c|} \hline & q_{\text{wait}} \\ \hline \pi_{\tilde{A}_1}(b) & \pi_{\tilde{A}}(c) \\ \hline \end{array}, \quad \pi_{\tilde{A}}(c) = \pi_{\tilde{A}_2}(b) = \pi_{\tilde{A}_1}(d) \text{ and } \pi_{\tilde{A}}(c) = \pi_{\tilde{A}_3}(e);$$

* si $\pi_{\mathcal{C}}(c) \neq \sim$, $\pi_{\mathcal{G}_1}(c) \in \{ \blacksquare, \blacktriangleright, \blacktriangleleft \}$ et $\pi_{\tilde{A}} = q_{\text{wait}}$, alors

$$\pi_{\tilde{A}}(c) = \pi_{\tilde{A}_3}(a) = \pi_{\tilde{A}_3}(e) = \pi_{\tilde{A}_2}(b) = \pi_{\tilde{A}_1}(d) = q_{\text{wait}},$$

* si $\pi_{\mathcal{C}}(c) \neq \sim$ et $\pi_{\tilde{A}_1}(b) = q_{\text{wait}}$ ou bien $\pi_{\tilde{A}_2}(d) = q_{\text{wait}}$ alors $\pi_{\tilde{A}}(c) = q_{\text{wait}}$.

Ces conditions de type fini permettent de définir le décalage sofique $\mathbf{T}_{\mathcal{M}}$ de la manière suivante :

$$\mathbf{T}_{\mathcal{M}} = \mathbf{TF}_{\text{Transfert} \cup \text{Init} \cup \text{Comp} \cup \text{Bord}} \left(\mathbf{Prod} \left(\mathbf{T}_{\text{Horloge}}, \tilde{A}^{\mathbb{Z}^2} \right) \right).$$

Pour plus de clarté, toutes les conditions de type fini **Transfert**, **Init**, **Comp** et **Bord** sont regroupées en une seule condition **Work_M**. La construction se résume ainsi en : $\mathbf{T}_{\mathcal{M}} = \mathbf{TF}_{\text{Work}_{\mathcal{M}}} \left(\mathbf{Prod} \left(\mathbf{T}_{\text{Grid}}, A_{\text{Comp}(\mathcal{M})}^{\mathbb{Z}^2} \right) \right)$ pour toute machine de Turing \mathcal{M} .

Sur chaque bande de calcul apparaissent ainsi des morceaux du diagramme espace-temps de la machine de Turing \mathcal{M} , avec pour mot d'entrée le mot vide. Chacun de ces morceaux de diagramme espace-temps est limité à la fois en

espace, par la taille de la bande de calcul, et en temps, borné exponentiellement en la taille de la bande de calcul. Il est donc possible de trouver dans le décalage sofique \mathbf{T}_M des fragments arbitrairement grands du diagramme espace-temps de la machine M , ce qui montre le résultat principal de cette partie :

Théorème 2.3.4. [AS10] *Pour toute machine de Turing M avec ruban semi-infini, il existe un décalage sofique \mathbf{T}_M qui contient les diagrammes espace-temps de taille $2^n \times (2^{2^n} + 2)$ du calcul de la machine M sur le mot vide, pour tout entier n (2^n tuiles de calcul et $2^{2^n} + 2$ étapes de calcul).*

Exemple 2.3.2. Dans cet exemple la machine M_{ex} commence son énumération par le mot ab . La Figure 17 décrit comment le calcul de la machine est codé dans la grille de calcul. Par exemple les tuiles de calcul de niveau 2 forment une zone de calcul de taille 4×18 (ruban à quatre cellules et dix-huit étapes de calcul).

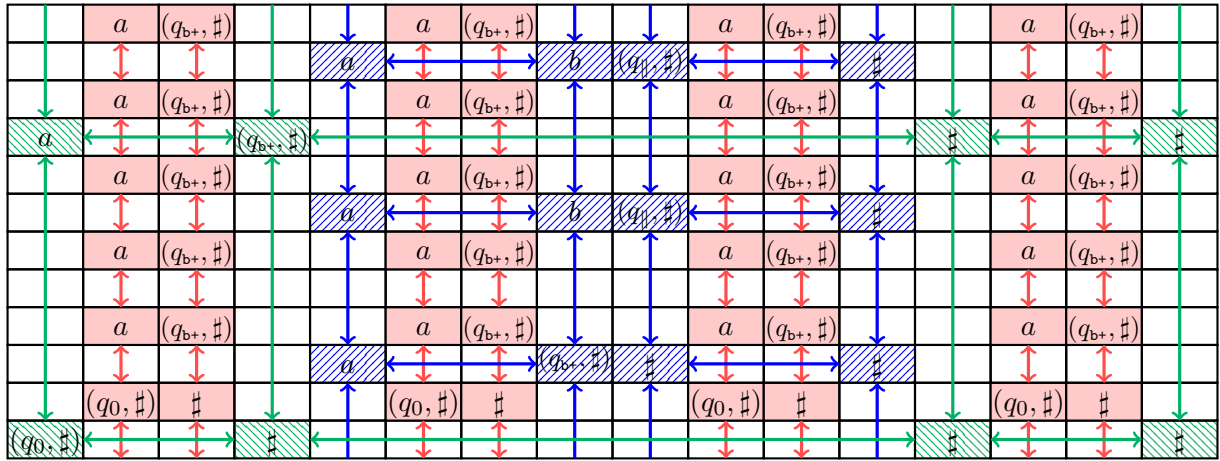


FIGURE 17: Calculs d'une machine de Turing sur une grille de calcul dans laquelle apparaissent des zones de calcul de niveaux 1, 2 et 3. Il est à noter que chaque symbole \uparrow ou \leftrightarrow devrait en fait transférer deux symboles, mais ceux-ci ont ici été omis pour plus de lisibilité. Pour la même raison les états de l'horloge n'apparaissent pas.

3 Simulation de décalages effectifs 1D par des décalages de type fini 2D

3.1	Un résultat de simulation	63
3.1.1	Énoncé du résultat et idée de la preuve	63
3.1.2	Communication entre machines	65
	Canaux de communication	65
	Adresses dans une bande de calcul	67
	Zones de responsabilité	69
3.1.3	Génération de motifs interdits	70
3.1.4	Détection de motifs interdits	72
3.1.5	Construction finale	76
3.2	Deux applications du théorème de simulation	77
3.2.1	Ordres sur les langages de motifs et son équivalent sur les décalages	78
	Un pré-ordre sur les langages	78
	Théorème de clôture	80
3.2.2	Décalages S-adiques multidimensionnels effectifs	86
	Les décalages S-adiques multidimensionnels effectifs sont sofiques	87
	Une réciproque?	90

Le principal résultat de ce chapitre est le Théorème 3.1.1, que nous appellerons théorème de simulation. Ce théorème ne concerne que deux des cinq opérations définies dans la Partie 2.2 : le facteur et la sous-action projective. Michael Hochman montre [Hoc09] que tout décalage effectif de dimension d peut être obtenu comme facteur et sous-action d'un décalage de type fini de dimension $d + 2$. Sa construction amène naturellement la question suivante : est-il possible de diminuer la dimension du décalage de type fini à $d + 1$? Une réponse positive a été apportée dans deux preuves faisant appel à des techniques

de pavages différentes. L'une [DRS10b] due à Bruno Durand, Andrei Romaschenko et Alexander Shen utilise une construction basée sur des pavages auto-similaires [DLS01]. Celle que nous proposons [AS10], qui est une adaptation de la preuve de Robinson [Rob71] de l'indécidabilité du problème du domino, dans laquelle les zones de calcul sont remplacées par des bandes de calcul bi-infinies verticalement. Cette preuve est détaillée dans la Partie 3.1. Les premiers auteurs proposent un comparatif des deux constructions dans [DRS10a].

Dans la Partie 3.2 nous développons deux applications du théorème de simulation. La première application (Théorème 3.2.2) établit une correspondance entre un semi-ordre sur les décalages et un semi-ordre sur les langages, pourvu que les décalages vérifient la condition de mélange uniforme. La seconde application concerne les systèmes S-adiques ; nous montrons qu'en dimension supérieure à 2, les systèmes S-adiques effectifs sont sofiques (Théorème 3.2.5).

3.1 Un résultat de simulation

Ce résultat de simulation a initialement été démontré par Hochman dans une version plus faible [Hoc09]. En terme de clôture, ce résultat s'exprime de la manière suivante :

$$Cl_{\text{Fact,SA}}(SFT) = \mathcal{RE},$$

autrement dit les décalages que l'on peut obtenir par facteur et sous-action à partir d'un décalage de type fini sont exactement les décalages effectifs (ces classes de décalages sont définies dans la Partie 1.3).

3.1.1 Énoncé du résultat et idée de la preuve

Le résultat dont nous allons présenter la preuve s'énonce de la manière suivante :

Théorème 3.1.1 ([AS10]). *Tout décalage effectif de dimension d peut être obtenu par facteur et sous-action d'un décalage de type fini de dimension $d + 1$.*

L'énoncé du théorème peut être affiné, le facteur utilisé étant en fait un facteur lettre à lettre. On obtient alors le corollaire suivant :

Corollaire 3.1.2 ([AS10]). *Tout décalage effectif de dimension d est conjugué à une sous-action d'un décalage de type fini de dimension $d + 1$.*

Nous allons présenter une preuve de ce résultat dans le cas particulier où $d = 1$, mais la construction se généralise à la dimension supérieure.

Nous commençons par donner les idées directrices de la démonstration. Soit $\Sigma \subset A_{\Sigma}^{\mathbb{Z}}$ un décalage effectif de dimension 1 sur l'alphabet A_{Σ} , alors il existe

une machine de Turing \mathcal{M} qui énumère les motifs interdits de Σ . Nous allons construire un SFT $\mathbf{T}_{\text{Final}}$ de dimension 2. Ce décalage sera constitué de différents niveaux, chaque niveau étant un SFT de dimension 2, et ces niveaux seront regroupés par des opérations **Prod**, pour n'obtenir qu'un seul décalage au final, et des conditions de type fini **TF** de façon que chaque niveau utilise l'information contenue dans les autres niveaux. De manière plus détaillée, les différents niveaux sont :

- **Niveau 1** : ce premier niveau contient le décalage $A_{\Sigma}^{\mathbb{Z}^2}$ auquel on ajoute la condition de type fini **Align** qui impose à une même colonne de contenir un unique symbole répété verticalement. Ainsi ce niveau contient une superposition de la même configuration $x \in A_{\Sigma}^{\mathbb{Z}}$ (que l'on appellera candidat) ;
- **Niveau 2** : ce niveau contient les zones de calcul utilisées par $\mathcal{M}_{\text{Forbid}}$ et $\mathcal{M}_{\text{Search}}$, munies d'une horloge (voir la Partie 2.3.4) ;
- **Niveau 3** : ce niveau contient les diagrammes espace-temps d'une machine de Turing $\mathcal{M}_{\text{Forbid}}$ qui énumère les motifs interdits de Σ et vérifie qu'aucun des motifs interdits de Σ n'apparaît dans la configuration x ;
- **Niveau 4** : ce niveau contient les diagrammes espace-temps d'une machine de Turing $\mathcal{M}_{\text{Search}}$ qui aide la machine $\mathcal{M}_{\text{Forbid}}$ dans sa phase de vérification.

La difficulté est donc d'assurer qu'aucun motif interdit de Σ n'apparaît dans la configuration candidate x . La grille de calcul du deuxième niveau, présentée dans la Partie 2.3.4, offre donc la possibilité à une machine de Turing d'effectuer des calculs dans des zones de calcul finies de taille arbitrairement grande. Pour vérifier que le candidat x est bien dans Σ , nous utiliserons une machine de Turing $\mathcal{M}_{\text{Forbid}}$ qui aura une double fonction : d'abord énumérer sur son ruban les motifs interdits de Σ , puis vérifier qu'aucun d'entre eux n'est présent dans une partie finie de la configuration x , appelée zone de responsabilité de la machine, qui dépend de la taille de la zone de calcul dans laquelle la machine évoluera. Mais le problème des espaces de calcul présentés dans la Partie 2.3.2 est qu'ils sont non connexes. En conséquence, le ruban sur lequel travaille la machine $\mathcal{M}_{\text{Forbid}}$ est un ruban fractionné, donc les motifs interdits qu'elle énumère sont eux-mêmes fractionnés. Comment alors vérifier qu'un motif interdit fractionné n'apparaît pas dans la configuration candidate x , qui est faite de cellules connexes ?

Pour ce faire nous faisons appel à une deuxième machine de Turing $\mathcal{M}_{\text{Search}}$. Cette machine $\mathcal{M}_{\text{Search}}$ reçoit des requêtes de la part de la machine $\mathcal{M}_{\text{Forbid}}$,

sous forme d'une adresse d'une lettre de la configuration candidate x dans la zone de responsabilité de $\mathcal{M}_{\text{Forbid}}$. Comme toute position de la configuration x est située dans une zone de calcul d'un certain niveau, il suffit à la machine $\mathcal{M}_{\text{Search}}$ à laquelle est formulée la requête de communiquer avec cette autre machine $\mathcal{M}_{\text{Search}}$. Cette communication entre machines $\mathcal{M}_{\text{Search}}$ sera décrite dans la Partie 3.1.4. En supposant que la machine $\mathcal{M}_{\text{Search}}$ remplit bien son rôle, elle répond à une requête par la lettre de la configuration x correspondante, et la machine $\mathcal{M}_{\text{Forbid}}$ peut ainsi vérifier la validité de la configuration x . Si un motif interdit de Σ est détecté dans x , alors la machine $\mathcal{M}_{\text{Forbid}}$ atteint un état spécial q_{stop} , dont la présence sera interdite dans les configurations du décalage final $\mathbf{T}_{\text{Final}}$. Cette construction, qui est résumée dans la Partie 3.1.5, assure que toutes les lignes $x_{\mathbb{Z} \times \{i\}}$ dans le décalage final codent, entre autre, des configurations de Σ . Il reste alors à appliquer deux opérations pour obtenir Σ : d'abord prendre la sous-action de $\mathbf{T}_{\text{Final}}$ sur $\mathbb{Z} \times \{0\}$, et ensuite effacer tous les symboles de l'alphabet de $\mathbf{T}_{\text{Final}}$ qui ne concernent pas Σ (ceux utilisés pour la construction des zones de calcul, ceux codant le comportement des machines de Turing, etc...) par un facteur lettre à lettre.

Nous allons donc dans un premier temps décrire comment les différentes machines $\mathcal{M}_{\text{Forbid}}$ et $\mathcal{M}_{\text{Search}}$ communiquent entre elles (Partie 3.1.2).

3.1.2 Communication entre machines

Canaux de communication

Les machines de Turing $\mathcal{M}_{\text{Forbid}}$ et $\mathcal{M}_{\text{Search}}$ vont avoir besoin de communiquer entre elles. Entre deux zones de calcul adjacentes de même niveau, la communication est aisée puisqu'on ne trouve que des tuiles de communication entre ces zones. Ainsi un bit d'information peut être échangé entre deux zones de calcul adjacentes de niveau n à chaque étape de calcul (voir la Partie 2.3.2). Mais entre deux bandes de calcul de niveaux différents le problème est plus difficile. Nous construisons dans cette partie des canaux de communication pour permettre à des machines évoluant dans des bandes de calcul de différents niveaux de communiquer entre elles. Pour construire ces canaux de communication, nous utilisons une substitution s_2 sur l'alphabet \mathcal{G}_2 . Cet alphabet et les règles de la substitution s_2 sont décrites dans la Figure 18.

Nous modifions donc le décalage $\mathbf{T}_{\text{Horloge}}$ défini dans la Partie 2.3.4 pour y ajouter ces canaux de communication. Le nouveau décalage est donc :

$$\mathbf{T}_{\text{Horloge}} = \mathbf{TF}_{\text{Count} \cup \text{Consist} \cup \text{Synchro}} \left(\mathbf{Prod} \left(\mathbf{T}_{s_1 \times s_2}, \mathcal{C}^{\mathbb{Z}^2} \right) \right),$$

où la substitution $s_1 \times s_2$ est définie sur l'alphabet produit $\mathcal{G}_1 \times \mathcal{G}_2$ et dont

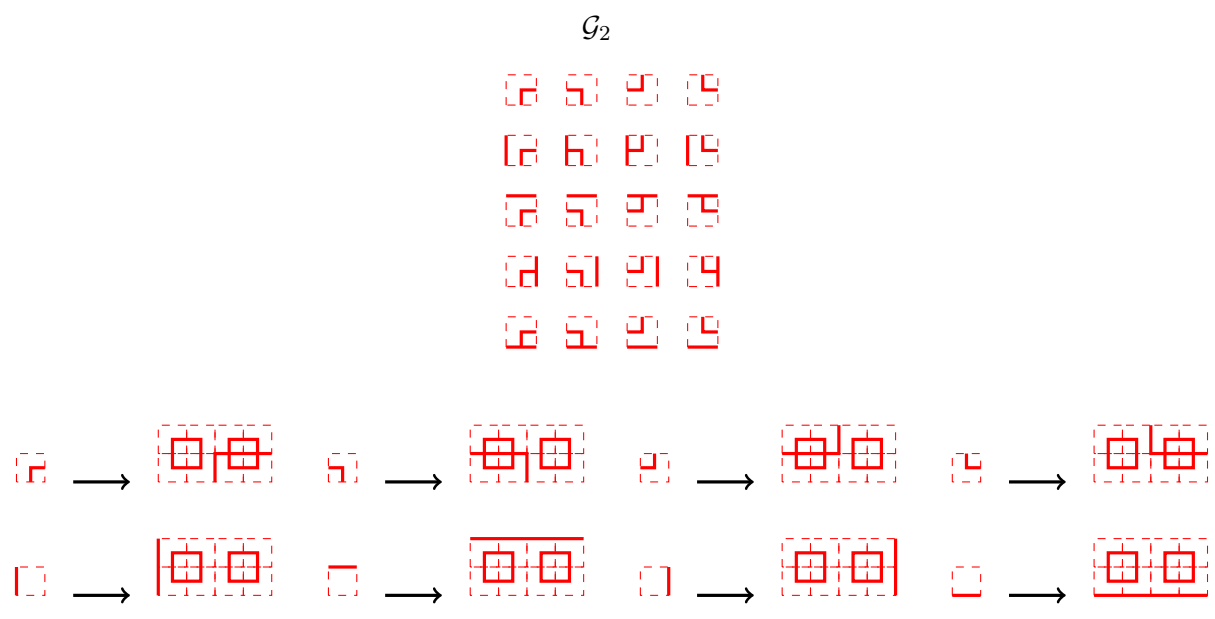


FIGURE 18: L'alphabet \mathcal{G}_2 et les règles de la substitution s_2 .

les règles sont formées d'une règle de s_1 et d'une règle de s_2 . En conséquence de quoi le décalage $\mathbf{T}_{\mathcal{M}}$ est lui aussi modifié pour intégrer ces canaux de communication :

$$\mathbf{T}_{\mathcal{M}} = \mathbf{TF}_{\text{Transfert} \cup \text{Init} \cup \text{Comp} \cup \text{Bord}} \left(\mathbf{Prod} \left(\mathbf{T}_{\text{Horloge}}, \tilde{A}^{\mathbb{Z}^2} \right) \right),$$

où le décalage $\mathbf{T}_{\text{Horloge}}$ est celui que l'on vient de redéfinir.

Les segments rouges obtenus après itération de la substitution s_2 sont appelées *lignes de communication*. Elles sont utilisées pour permettre la communication entre des bandes de calcul de différents niveaux. Les lignes de communication sont agencées de manière à former des rectangles. Les deux rectangles obtenus après n itérations de la substitution s_2 sur un élément de l'alphabet \mathcal{G}_2 sont appelés *rectangles de communication de niveau n* . Chaque rectangle de niveau n croise deux rectangles de niveau $n - 1$ et est intersecté par un rectangle de niveau $n + 1$.

Si l'on considère une tuile de calcul de bord \blacksquare (resp. \blacktriangleleft) dans une zone de calcul de niveau n , elle est toujours située à l'intérieur d'un rectangle de communication de niveau n . Ainsi en partant de la tuile de calcul \blacksquare (resp. \blacktriangleleft)

et en se déplaçant vers la gauche (resp. la droite), on rencontre nécessairement le côté gauche (resp. le côté droit) d'un de ces rectangles. Et sur les côtés supérieur et inférieur de ce rectangle se trouvent deux tuiles de calcul \Rightarrow et \Leftarrow , situées dans deux zones de calcul adjacentes de niveau $n - 1$.

A l'aide de règles locales il est possible de construire des *canaux de communications de niveau n* , qui partent de chaque tuile de calcul \Rightarrow ou \Leftarrow de niveau n . Un canal de communication est constitué d'une section horizontale qui rejoint un rectangle de communication de niveau n comme expliqué précédemment, et longe le bord de ce rectangle jusqu'à rencontrer une autre tuile de calcul de bord, de niveau $n - 1$. Ainsi une zone de calcul peut communiquer avec les quatre zones de calcul de niveau inférieur qu'elle contient (voir la Figure 19). Comme les zones de calcul de niveau n sont superposées verticalement avec une fréquence moitié de celle des zones de niveau $n - 1$, seule une moitié des zones de niveau $n - 1$ est connecté *via* un canal de communication à une zone de niveau supérieur.

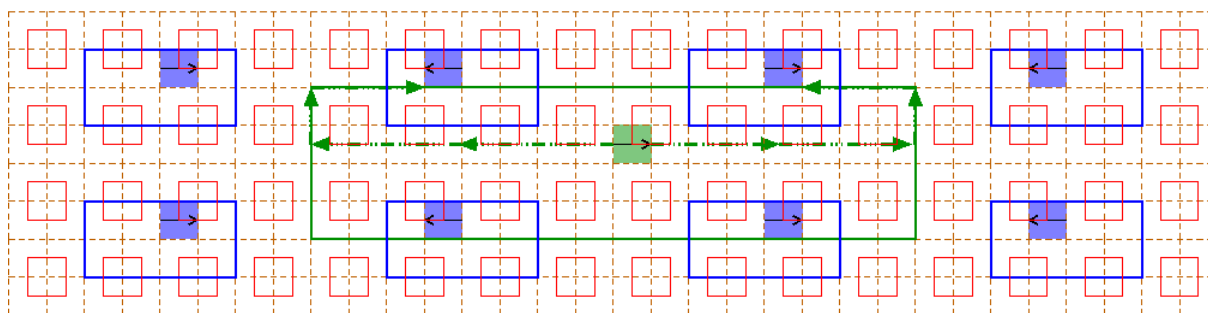


FIGURE 19: Les rectangles de communication permettent à chaque zone de calcul de niveau n de communiquer avec les quatre zones de calcul de niveau $n - 1$ les plus proches.

Proposition 3.1.3. *Pour toute zone de calcul de niveau n , il existe deux canaux de communication partant de chaque tuile de calcul \Rightarrow ou \Leftarrow de niveau n et terminant sur une tuile de calcul de bord de niveau $n - 1$ (une tuile \Rightarrow et une tuile \Leftarrow). Ainsi chaque zone de calcul de niveau n peut communiquer avec les quatre bandes de calcul de niveau $n - 1$ les plus proches.*

Adresses dans une bande de calcul

Dans cette partie nous expliquons comment définir une adresse permettant d'identifier chaque lettre x_i de la configuration candidate x située à l'intérieur

d'une bande de calcul. Comme par construction du premier niveau, la configuration x est répétée verticalement, il suffit de définir une adresse qui identifie la $i^{\text{ième}}$ colonne dans le décalage de type fini du premier niveau.

Soit C_n une zone de calcul de niveau n d'une configuration $x \in \mathbf{T}_{\text{Grid}}$ et soit S_n la bande de calcul associée. Par la Proposition 2.3.1, il existe un unique $i \in [0, 4^n - 1] \times [0, 2^n - 1]$ et un unique $y \in \mathbf{T}_{\text{Grid}}$ tels que $s_{\text{Grid}}^n(y) = \sigma^i(x)$. Donc il existe un unique $(j_1, j_2) \in \mathbb{Z}^2$ tel que $C_n \subset s_{\text{Grid}}^n(y_{\{j_1, j_2\}})$. On a $S_n \subset \sigma^{-i}(s_{\text{Grid}}^n(y_{\{j_1, j_2\}})) \subset x$, et la bande $\sigma^{-i}(s_{\text{Grid}}^n(y_{\{j_1, j_2\}}))$ est appelée *bande de dépendance* associée à la bande de calcul S_n .

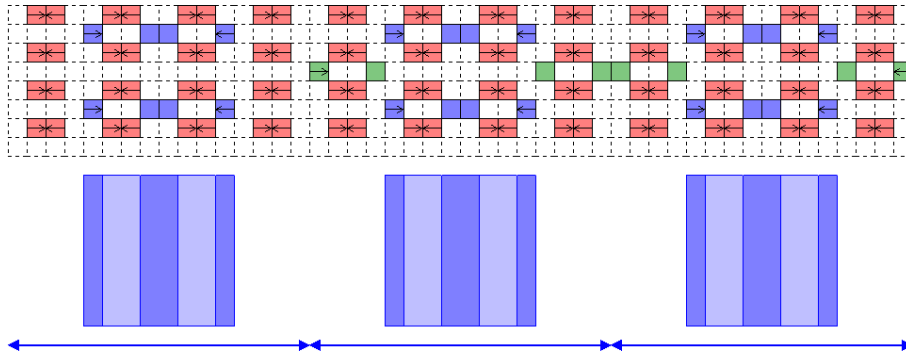


FIGURE 20: Les bandes de dépendance associée à des zones de calcul de niveau 2.

Dans le décalage \mathbf{T}_{Grid} , le ruban d'une machine de Turing dans une bande de calcul de niveau n est fractionné. De ce fait, une machine de Turing de niveau n ne peut pas accéder directement à toutes les colonnes qui composent sa bande de dépendance. Pour obtenir ces informations, la machine communique avec une machine de Turing de niveau inférieur (voir la Partie 3.1.4) mais il faut pour cela que les deux machines puissent identifier précisément de quelle colonne il est question.

Étant donnée une bande de dépendance associée à une bande de calcul de niveau n , il est possible d'expliciter des coordonnées relatives à cette bande pour chaque colonne. Ces adresses sont composées de n symboles choisis parmi l'alphabet à quatre éléments $\{0, 1, 2, 3\}$. Chacun des motifs $s_1^n(a)$ peut être décomposé horizontalement en quatre (éventuellement différents) motifs $s_1^{n-1}(b)$ où a, b sont des lettres de l'alphabet \mathcal{G}_1 . Le premier symbole de l'adresse indique à laquelle des bandes de dépendances de niveau $n - 1$ la colonne appartient. En itérant ce procédé, la position d'une colonne dans une bande de dépendance de niveau n est donnée par un mot de n lettres sur l'alphabet $\{0, 1, 2, 3\}$, que l'on

appelle *adresse de la colonne* (voir la Figure 21).

0				1				2				3				s_1^n
0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	s_1^{n-1}
																s_1^{n-2}

FIGURE 21: Adresses de deux colonnes dans une bande de dépendance de niveau 3. L'adresse de la colonne repérée par une tuile noire est 231, tandis que l'adresse de la colonne repérée par une tuile grise est 020.

Proposition 3.1.4. *Pour chaque zone de dépendance de niveau n , il est possible de décrire la position de chaque colonne de la bande par une adresse de n bits sur un alphabet à quatre éléments.*

Zones de responsabilité

Rappelons que la machine de Turing $\mathcal{M}_{\text{Forbid}}$, qui sera décrite plus précisément dans la Partie 3.1.4, a un double rôle : elle énumère les motifs interdits de Σ et vérifie que ces motifs n'apparaissent pas dans la configuration x . Mais vérifier l'absence d'un motif interdit dans l'intégralité de la configuration x nécessite une infinité d'étapes de calcul, c'est pourquoi chacune des machines $\mathcal{M}_{\text{Forbid}}$ ne se charge de la vérification que d'une partie finie de la configuration x . Cette partie finie est appelée *zone de responsabilité* de la machine $\mathcal{M}_{\text{Forbid}}$.

Nous assignons donc une zone de responsabilité à chaque bande de calcul. Pour une bande de calcul de niveau n cette zone de responsabilité est formée de $3 * (2 * 4^{n-1}) = 6 * 4^{n-1}$ cellules, centrées autour de la bande de calcul (voir la Figure 22), de sorte que la zone de responsabilité d'une machine de niveau n débute là où la zone de responsabilité de sa voisine de gauche de même niveau termine.

Ainsi définies, les zones de responsabilité se chevauchent : deux zones de responsabilité adjacentes de même niveau n partagent $2 * 4^{n-1}$ cellules. Ces chevauchements sont essentiels : sans eux, on pourrait imaginer un motif interdit à cheval sur deux zones de responsabilité, et qui ne serait ainsi jamais détecté. Pour la même raison il est important que la taille des chevauchements croisse avec le niveau des machines, cela nous assure que tout motif de la configuration candidate x est contenu dans une infinité de zones de responsabilité de niveaux croissants.

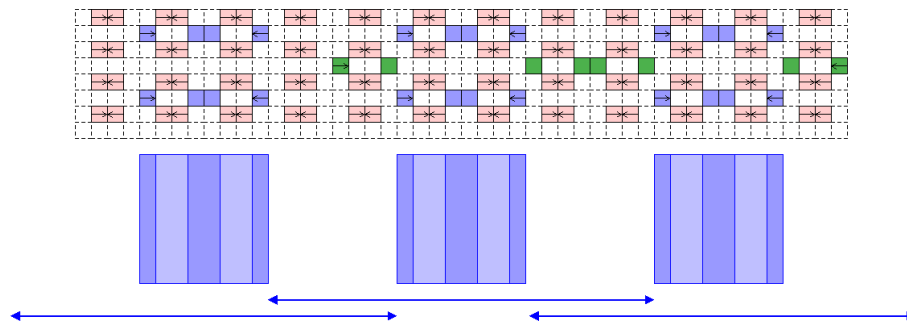


FIGURE 22: Zones de responsabilité pour des bandes de niveau 2. Ces zones sont formées de 24 cellules et se chevauchent sur 8 cellules.

3.1.3 Génération de motifs interdits

Rappelons que comme le décalage Σ est récursivement énumérable, il existe une machine de Turing qui énumère ses motifs interdits. Nous décrivons dans cette partie une version modifiée de cette machine, qui en plus de l'énumération vérifie que chaque motif interdit qu'elle produit n'apparaît pas dans sa zone de responsabilité, les comparaisons se faisant avec les symboles du décalage du premier niveau. Les zones de calcul sont fractionnées (voir la Figure 17), donc durant un calcul sur une bande de calcul de taille 2^n , une machine $\mathcal{M}_{\text{Forbid}}$ ne peut pas accéder entièrement à sa zone de responsabilité. La machine $\mathcal{M}_{\text{Forbid}}$ sollicitera l'aide d'une deuxième machine de Turing $\mathcal{M}_{\text{Search}}$ afin d'obtenir les symboles de la configuration candidate $x \in A_{\Sigma}^{\mathbb{Z}}$ situés à l'intérieur de sa zone de responsabilité, et dont elle aura besoin. Le comportement de la machine $\mathcal{M}_{\text{Forbid}}$ est le suivant : elle énumère autant de motifs interdits de Σ que la taille de sa zone de calcul le lui permet, et chaque fois qu'un motif est produit, la machine $\mathcal{M}_{\text{Forbid}}$ vérifie que ce motif n'apparaît pas dans sa zone de responsabilité. Nous allons décrire ce fonctionnement plus précisément, en commençant par présenter les trois rubans dont dispose la machine $\mathcal{M}_{\text{Forbid}}$.

Rubans de la machine $\mathcal{M}_{\text{Forbid}}$ La machine $\mathcal{M}_{\text{Forbid}}$ utilise trois rubans :

- le premier ruban est le *ruban de calcul* ;
- le deuxième ruban est le *ruban d'écriture*, sur lequel les motifs interdits sont écrits les uns après les autres ;
- le troisième et dernier ruban est le *ruban de communication*, qui contient les adresses (une adresse remplaçant la précédente) des lettre de l'alphabet

A_Σ dont la machine $\mathcal{M}_{\text{Forbid}}$ a besoin pour vérifier que le motif écrit sur son ruban d'écriture n'apparaît pas dans sa zone de responsabilité. La machine $\mathcal{M}_{\text{Forbid}}$ attend que la machine $\mathcal{M}_{\text{Search}}$, choisie parmi les trois machines $\mathcal{M}_{\text{Search}}$ de son voisinage (à sa gauche, au centre ou à sa droite), qu'elle a sollicitée soit disponible. Puis lui transfère l'adresse de la lettre de A_Σ dont elle a besoin (voir la Partie 3.1.4).

Nous développons à présent les différentes étapes d'un calcul de la machine $\mathcal{M}_{\text{Forbid}}$.

Calcul de la taille de la zone de responsabilité La machine $\mathcal{M}_{\text{Forbid}}$ commence par calculer la taille de sa zone de calcul, comprise entre les tuiles de calcul \blackrightarrow et \blackleftarrow . De cette façon, la machine $\mathcal{M}_{\text{Forbid}}$ connaît la taille de sa zone de responsabilité, et ceci peut être effectué en temps linéaire par rapport à la taille de la zone de calcul.

Énumération de motifs interdits Une fois le calcul précédent effectué, la machine $\mathcal{M}_{\text{Forbid}}$ énumère les motifs interdits de Σ , et dès qu'un motif est intégralement inscrit sur son ruban d'écriture, elle vérifie que ce motif n'apparaît pas à l'intérieur de sa zone de responsabilité.

Vérification de la zone de responsabilité Supposons que la machine $\mathcal{M}_{\text{Forbid}}$ a inscrit sur son ruban d'écriture le motif interdit $f = f_0 f_1 \dots f_{k-1}$, et que la machine $\mathcal{M}_{\text{Forbid}}$ est chargée d'une zone de responsabilité de niveau n que l'on appellera $a_0 a_1 \dots a_{6 \cdot 4^{n-1} - 1}$. Elle commence par demander à une des machines $\mathcal{M}_{\text{Search}}$ voisines la lettre a_0 dont l'adresse est $0 \dots 0$ où le symbole 0 est répété n fois (le principe d'une telle requête sera développé dans la Partie 3.1.4), et compare la lettre a_0 avec la première lettre du motif interdit f_0 . Si les lettres sont identiques, il est encore possible que le motif interdit f apparaisse en position 0 dans la zone de responsabilité, donc la machine garde en mémoire la fait qu'il faille continuer la comparaison des motifs f et $a_0 \dots a_k$. Si $f_0 \neq a_0$ on est sûr que le motif interdit f n'apparaît pas à cette position. Dans les deux cas, la machine $\mathcal{M}_{\text{Forbid}}$ fait à nouveau appel à une machine $\mathcal{M}_{\text{Search}}$ pour obtenir le deuxième symbole de sa zone de responsabilité a_1 d'adresse $0 \dots 01$ où le symbole 0 est répété $n - 1$ fois. Si cela est pertinent, elle compare a_1 avec f_1 . Puis elle compare a_1 avec f_0 , et ainsi de suite. Si à un moment la machine $\mathcal{M}_{\text{Forbid}}$ détecte que $f = a_i \dots a_{i+k+1}$ pour un certain i , elle arrête tout calcul et entre dans un état q_{stop} dans lequel elle restera jusqu'à la réinitialisation de l'horloge. Cet état q_{stop} sera interdit dans le décalage final. Dans le pire des cas (si aucun motif interdit n'est détecté) pour chaque motif interdit

f de longueur k , la machine $\mathcal{M}_{\text{Forbid}}$ a besoin de $6 * 4^{n-1} * k * t(n)$ étapes de calcul, où $t(n)$ est le temps nécessaire à la machine $\mathcal{M}_{\text{Search}}$ pour répondre à une requête de $\mathcal{M}_{\text{Forbid}}$. Dans la Partie 3.1.4 nous donnerons une estimation de ce temps de réponse.

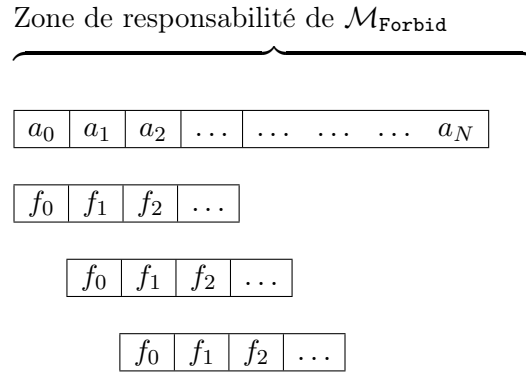


FIGURE 23: Lorsqu'un motif interdit de $\Sigma f = f_0 f_1 \dots f_k$ est énuméré par la machine $\mathcal{M}_{\text{Forbid}}$, la recherche de ce motif à toutes les positions possibles de la zone de responsabilité de $\mathcal{M}_{\text{Forbid}}$ est effectuée en parallèle.

3.1.4 Détection de motifs interdits

La machine de Turing $\mathcal{M}_{\text{Search}}$ reçoit une *requête*, c'est-à-dire une suite de symboles codant l'adresse d'une lettre à l'intérieur d'une zone de responsabilité d'une machine $\mathcal{M}_{\text{Forbid}}$, de la part d'une machine $\mathcal{M}_{\text{Forbid}}$ chaque fois qu'une adresse est intégralement inscrite sur le ruban de communication de la machine $\mathcal{M}_{\text{Forbid}}$. La machine $\mathcal{M}_{\text{Search}}$ doit renvoyer la lettre correspondante sur le premier niveau de construction $A_{\Sigma}^{\mathbb{Z}^2}$. Il faut remarquer que la zone de responsabilité d'une machine $\mathcal{M}_{\text{Forbid}}$ de niveau n ne correspond pas tout à fait à l'ensemble des positions qu'une machine $\mathcal{M}_{\text{Search}}$ de même niveau peut atteindre en faisant appel aux machines d'ordre inférieur. En fait une machine $\mathcal{M}_{\text{Forbid}}$ partage sa zone de responsabilité avec trois machines $\mathcal{M}_{\text{Search}}$, et selon l'adresse du symbole recherché, la machine $\mathcal{M}_{\text{Forbid}}$ envoie sa requête à la machine $\mathcal{M}_{\text{Search}}$ idoïne (voir la Figure 24 pour un exemple).

Décrivons plus précisément le comportement d'une machine $\mathcal{M}_{\text{Search}}$, en commençant par détailler les cinq rubans qu'elle utilise.

Rubans d'une machine $\mathcal{M}_{\text{Search}}$ Une machine $\mathcal{M}_{\text{Search}}$ de niveau n utilise trois rubans :

- le premier ruban est le *ruban de calcul* ;
- le deuxième ruban est le *ruban de requête d'ordre supérieur*, c'est sur ce ruban que les bits d'une adresse transférée par une machine $\mathcal{M}_{\text{Search}}$ de niveau $n + 1$ sont écrits.
- les trois derniers rubans sont le *ruban de requête gauche*, le *ruban de requête central* et le *ruban de requête droite* sur lesquels seront inscrits les adresses des symboles requis par la machine $\mathcal{M}_{\text{Forbid}}$ de niveau n située respectivement à la gauche, au même endroit et à la droite de la bande de calcul de la machine $\mathcal{M}_{\text{Search}}$ considérée.

Requête envoyée par une machine $\mathcal{M}_{\text{Forbid}}$ A chaque fois qu'une adresse est écrite sur le ruban de communication d'une machine $\mathcal{M}_{\text{Forbid}}$, celle-ci envoie cette requête à la machine $\mathcal{M}_{\text{Search}}$ de même niveau située soit dans la même bande de calcul, soit dans la bande de calcul directement à gauche ou directement à droite. La machine $\mathcal{M}_{\text{Forbid}}$ envoie les bits formant l'adresse un par un (un bit est envoyé à chaque étape de calcul), et donc une machine de niveau n envoie un bit toutes les 2^n lignes (avec l'implémentation des machines de Turing présentée dans la Partie 2.3). Deux bandes de calcul adjacentes de même niveau peuvent communiquer grâce aux canaux de communication décrits dans la Partie 3.1.2, en utilisant le fait qu'une ligne ne comprend que des zones de calcul de même niveau (voir la Proposition 2.3.2). Les bits de l'adresse étant envoyés un à un, le transfert de la totalité de l'adresse nécessite $2^n * n$ lignes. La requête est alors écrite sur le ruban de requête correspondant. La machine $\mathcal{M}_{\text{Forbid}}$ attend ensuite la réponse de la machine $\mathcal{M}_{\text{Search}}$ correspondante avant de poursuivre son calcul.

Requête envoyée par une machine $\mathcal{M}_{\text{Search}}$ Une machine $\mathcal{M}_{\text{Search}}$ de niveau $n \geq 2$ peut envoyer sa requête à l'une des quatre machines $\mathcal{M}_{\text{Search}}$ de niveau $n - 1$ situées dans sa zone de dépendance. De manière similaire à ce qui se passe pour une machine $\mathcal{M}_{\text{Forbid}}$, l'envoi des n bits composant l'adresse se fait bit par bit et nécessite donc $2^n * n$ ligne de calcul. La machine $\mathcal{M}_{\text{Search}}$ de niveau n utilise elle aussi les canaux de communication décrits dans la Proposition 3.1.3 pour communiquer : chaque tuile de calcul de bord \rightarrow ou \leftarrow est située à l'intérieur d'un rectangle de niveau n qui leur permet de communiquer avec une tuile de calcul de bord d'une zone de calcul de niveau $n - 1$.

Traitement d'une requête : l'arbre de recherche Une machine $\mathcal{M}_{\text{Search}}$ de niveau n répond aux requêtes écrites sur ses différents ruban, en consacrant une étape de calcul sur quatre à chacun de ses rubans. L'adresse inscrite sur le ruban est recopiée sur le ruban de calcul, et la machine garde en mémoire de quel ruban provient la requête qu'elle est en train de traiter. Si la machine $\mathcal{M}_{\text{Search}}$ est une machine de niveau 1, elle peut directement lire la lettre de A_Σ . Sinon elle est de niveau n , et transmet l'adresse à la machine $\mathcal{M}_{\text{Search}}$ de niveau $n - 1$: le premier bit de l'adresse indique quel canal de communication utiliser, et seuls les $n - 1$ derniers bits sont transmis. Puis la machine $\mathcal{M}_{\text{Search}}$ de niveau n attend la réponse à cette requête. Dès qu'une machine de niveau 1 reçoit la requête le symbole correspondant dans A_Σ est trouvé (voir la Figure 24). Il ne reste plus qu'à remonter l'arbre de recherche pour faire parvenir le symbole à la machine $\mathcal{M}_{\text{Forbid}}$ qui avait initialement formulé la requête.

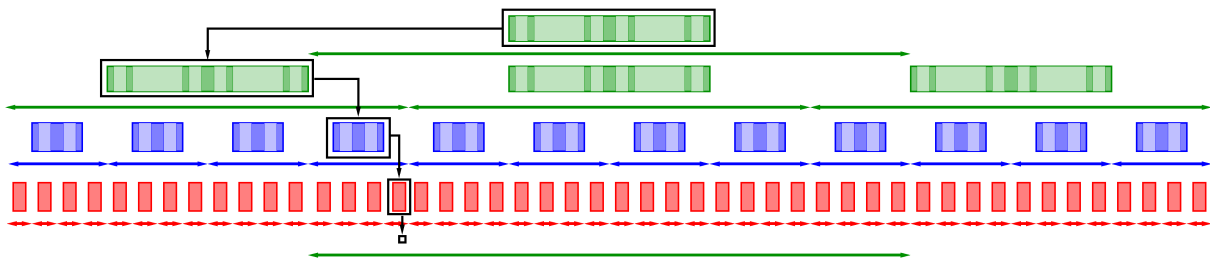


FIGURE 24: Un exemple d'arbre de recherche correspondant à une requête par une machine $\mathcal{M}_{\text{Forbid}}$ de niveau 3. Selon l'adresse de la lettre requise, la machine $\mathcal{M}_{\text{Forbid}}$ envoie la requête à une des trois machines $\mathcal{M}_{\text{Search}}$.

Réponse à une requête : remonter l'arbre de recherche Lorsqu'une machine $\mathcal{M}_{\text{Search}}$ reçoit de la machine de niveau inférieur le symbole correspondant à la requête qu'elle lui avait formulé, elle le transfère par le même canal de communication que celui d'où lui est venu cette requête. Cette opération se fait en une seule étape de calcul pour deux raisons. D'abord parce qu'il n'y a qu'un seul symbole à transmettre, mais aussi parce que la machine à laquelle ce symbole est envoyé attend cette réponse et donc sait de quel symbole il s'agit. Une machine $\mathcal{M}_{\text{Search}}$ finit toujours par répondre à une requête d'une machine $\mathcal{M}_{\text{Forbid}}$ partageant la même bande de calcul, puisque chacune des $\mathcal{M}_{\text{Search}}$ travaille à tour de rôle pour les machines $\mathcal{M}_{\text{Forbid}}$ de même niveau et la machine $\mathcal{M}_{\text{Search}}$ de niveau supérieur.

Initialisation des calculs Lorsque l'horloge est réinitialisée, il est crucial de ne pas effacer les adresses écrites sur les rubans de requêtes des machines $\mathcal{M}_{\text{Search}}$, car des machines de niveau supérieur sont potentiellement en train d'attendre une réponse.

Un autre problème pouvant survenir lors d'une réinitialisation est la suivante : une machine a envoyé une requête, et elle est réinitialisée alors qu'elle attend toujours la réponse à sa requête. Pour parer à cette difficulté on impose que la première chose que fasse une machine lorsqu'elle est réinitialisée est d'attendre la réponse à une éventuelle requête laissée en suspens dans son calcul précédent, et d'ignorer cette réponse avant de recommencer un nouveau calcul.

Temps de réponse d'une machine $\mathcal{M}_{\text{Search}}$ On appelle $t(n)$ le temps nécessaire à une machine $\mathcal{M}_{\text{Search}}$ de niveau n pour répondre à une requête d'une machine $\mathcal{M}_{\text{Forbid}}$ de même niveau. Comme une machine de niveau n effectue une étape de calcul toutes les 2^n lignes, une machine $\mathcal{M}_{\text{Search}}$ de niveau n a besoin de $2^n * t(n)$ lignes pour répondre à une requête d'une machine $\mathcal{M}_{\text{Forbid}}$.

Une machine $\mathcal{M}_{\text{Search}}$ de niveau $n \geq 2$ est aidée par une machine $\mathcal{M}_{\text{Search}}$ de niveau $n - 1$: elle lui transfère un à un les $n - 1$ derniers bits de l'adresse de la requête (pour cela elle a besoin de $n * 2^n$ lignes). Puis elle attend la réponse de la machine $\mathcal{M}_{\text{Search}}$ de niveau $n - 1$. Il est possible que cette machine $\mathcal{M}_{\text{Search}}$ de niveau $n - 1$ soit déjà en train de répondre à d'autres requêtes, dans le pire des cas trois autres requêtes. Au total une machine $\mathcal{M}_{\text{Search}}$ de niveau $n - 1$ peut être amenée à répondre à quatre requêtes simultanément. Au final, le temps de réponse d'une machine $\mathcal{M}_{\text{Search}}$ de niveau n est donné par la formule de récurrence :

$$2^n t(n) \leq n * 2^n + 4 * 2^{n-1} * t(n - 1).$$

De l'inégalité précédente nous déduisons que

$$t(n) \leq 2^n * \mathcal{O}(n^2 2^n).$$

Ainsi une machine $\mathcal{M}_{\text{Search}}$ de niveau n répond à une requête d'une machine $\mathcal{M}_{\text{Forbid}}$ en $\mathcal{O}(n^2 2^n)$ étapes de calcul.

Proposition 3.1.5. *Toutes les requêtes d'une machine $\mathcal{M}_{\text{Forbid}}$ de niveau n reçoivent une réponse d'une machine $\mathcal{M}_{\text{Search}}$ de même niveau en au plus $n^2 2^n$ étapes de calcul, pour un n assez grand.*

Temps de vérification par $\mathcal{M}_{\text{Forbid}}$ Supposons qu'une machine $\mathcal{M}_{\text{Forbid}}$ ait énuméré sur son ruban un motif f de taille k . Elle doit à présent s'assurer que ce motif n'apparaît pas dans sa zone de responsabilité. D'après ce qui a été

présenté dans la Partie 3.1.4, ceci nécessite $6 * 4^{n-1} * k * t(n) \leq k * n^2 * 2^{3n+1}$ étapes de calcul.

Soit $(f_i)_{i \in \mathbb{N}}$ l'énumération des motifs interdits de Σ faite par la machine $\mathcal{M}_{\text{Forbid}}$. On appelle $t(f_0, \dots, f_k)$ le temps nécessaire à la machine $\mathcal{M}_{\text{Forbid}}$ pour vérifier que les motifs $(f_i)_{i \in [0, k]}$ n'apparaissent pas dans sa zone de responsabilité, et $t'(f_0, \dots, f_k)$ le temps nécessaire à cette même machine pour uniquement énumérer ces motifs sans vérifier leur absence dans sa zone de responsabilité. Le temps nécessaire à une machine $\mathcal{M}_{\text{Forbid}}$ de niveau n pour vérifier l'absence des motifs interdits $(f_i)_{i \in [0, k]}$ est donc

$$t(f_0, \dots, f_k) \leq t'(f_0, \dots, f_k) + (k + 1) * \max\{|f_i| : i \in [0, k]\} * n^2 * 2^{3n+1}.$$

Comme $t'(f_0, \dots, f_k)$ ne dépend pas du niveau de la machine $\mathcal{M}_{\text{Forbid}}$, et qu'une machine $\mathcal{M}_{\text{Forbid}}$ de niveau n peut effectuer $2^{2^n} + 2$ étapes de calcul (voir le Théorème 2.3.4), il existe un niveau n tel que toutes les machines de ce niveau vérifient que les motifs interdits $(f_i)_{i \in [0, k]}$ n'apparaissent pas dans leurs zones de responsabilité.

Proposition 3.1.6. *Pour tout motif interdit de Σ , il existe un entier $n \in \mathbb{N}$ tel que dans tout calcul dans une bande de niveau n , le mot w est énuméré par la machine $\mathcal{M}_{\text{Forbid}}$ et celle-ci a le temps de vérifier qu'il n'apparaît pas dans sa zone de responsabilité.*

3.1.5 Construction finale

Nous résumons la construction finale du décalage $\mathbf{T}_{\text{Final}}$ étape par étape.

1. Tout d'abord nous construisons les quatre niveaux du décalage :

$$\mathbf{T}_{\text{Level}} = \mathbf{Prod} \left(A^{\mathbb{Z}^2}, \mathbf{T}_{\text{Grid}}, A_{\text{Comp}(\mathcal{M}_{\text{Forbid}})}^{\mathbb{Z}^2}, A_{\text{Comp}(\mathcal{M}_{\text{Search}})}^{\mathbb{Z}^2} \right).$$

2. Ensuite, nous alignons verticalement les lettres du premier niveau pour obtenir la même configuration candidate sur chaque ligne :

$$\mathbf{T}_{\text{Align}} = \mathbf{TF}_{\text{Align}}(\mathbf{T}_{\text{Level}}).$$

3. Puis nous insérons les calculs des machines $\mathcal{M}_{\text{Forbid}}$ et $\mathcal{M}_{\text{Search}}$ à l'aide des conditions de type fini $\mathbf{Work}_{\mathcal{M}_{\text{Forbid}}} \cup \mathbf{Work}_{\mathcal{M}_{\text{Search}}}$, et nous y ajoutons les communications entre les différents niveaux de la construction, toujours avec des conditions de type fini **Com**. Enfin nous finissons par la condition **Forbid** qui permet d'éliminer les configurations dans lesquelles une machine $\mathcal{M}_{\text{Forbid}}$ détecte un motif interdit. Nous obtenons alors :

$$\mathbf{T}_{\text{Final}} = \mathbf{TF}_{\mathbf{Work}_{\mathcal{M}_{\text{Forbid}}} \cup \mathbf{Work}_{\mathcal{M}_{\text{Search}}} \cup \mathbf{Com} \cup \mathbf{Forbid}}(\mathbf{T}_{\text{Align}}).$$

Appelons \mathbf{T} le décalage $\mathbf{Fact}_\pi(\mathbf{SA}_{(1,0)\mathbb{Z}}(\mathbf{T}_{\mathbf{Final}}))$, où π est le facteur qui consiste à ne conserver que les lettres de l'alphabet A_Σ du premier niveau. Montrons que les décalages Σ et \mathbf{T} sont en fait identiques.

Démonstration. • $\Sigma \subseteq \mathbf{T}$:

Soit $x \in \Sigma$, par construction du décalage $\mathbf{T}_{\mathbf{Final}}$ il est facile de construire une configuration bidimensionnelle y telle que $y \in \mathbf{T}_{\mathbf{Final}}$ et $\pi(y|_{(1,0)\mathbb{Z}}) = x$.

• $\mathbf{T} \subseteq \Sigma$:

Soit $x \in \mathbf{T}$, nous montrons que $x \in \Sigma$. Par définition, il existe $y \in \mathbf{T}_{\mathbf{Final}}$ tel que $\pi(y|_{\mathbb{Z}e_1}) = x$. Il nous suffit de montrer que tout motif qui apparaît dans x est dans le langage $\mathcal{L}(\Sigma)$. Soit w un motif qui apparaît dans x . Supposons que w n'est pas dans $\mathcal{L}(\Sigma)$. Alors par la Proposition 3.1.6, il existe un $n \in \mathbb{N}$ tel que dans tout calcul dans une bande de niveau n , le mot w est énuméré par la machine $\mathcal{M}_{\mathbf{Forbid}}$ et celle-ci a eu le temps de vérifier qu'il n'apparaît pas dans sa zone de responsabilité. En particulier le motif w a été comparé avec tous les autres motifs de même longueur qui apparaissent dans x . Comme w apparaît dans x , cela signifie qu'il existerait une bande de calcul de niveau n dans laquelle la condition de type finie **Forbid** n'est pas respectée, d'où la contradiction. \square

3.2 Deux applications du théorème de simulation

Nous présentons dans cette partie deux résultats impliquant des décalages sur \mathbb{Z}^d . Ces deux résultats utilisent le théorème de simulation, mais le premier constitue plus une adaptation de la preuve qu'une application à proprement parler.

Le théorème de simulation peut aussi être utilisé pour montrer que les valeurs possibles pour l'entropie d'un décalage de type fini sur \mathbb{Z}^d sont les réels que l'on peut approcher par le haut à l'aide d'une suite calculable de rationnels [HM10]. Une autre application montre l'existence d'un jeu de tuiles tel que tout pavage quasi-périodique par ce jeu de tuiles a une fonction de quasi-périodicité non récursivement bornée [BJ10].

3.2.1 Ordres sur les langages de motifs et son équivalent sur les décalages

Dans la Partie 2.2 nous avons présenté des opérations sur les décalages, ainsi que quelques résultats de stabilité de classes par ces opérations. Une question qui peut venir à l'esprit est de savoir si, étant donné un décalage \mathbf{T} , on peut caractériser l'ensemble des décalages que l'on peut obtenir en appliquant ces opérations. Autrement dit, peut-on caractériser l'ensemble $Cl_{\mathbf{Prod}, \mathbf{Fact}, \mathbf{TF}, \mathbf{SA}, \mathbf{SE}}(\mathbf{T})$? Pour répondre à cette question, nous introduisons la notion de machine de Turing avec semi-oracle, qui est un cas particulier de machine de Turing avec oracle [Rog87]. Nous montrons qu'au semi-ordre sur les décalages $\leq_{\mathbf{Prod}, \mathbf{Fact}, \mathbf{TF}, \mathbf{SA}, \mathbf{SE}}$ correspond un semi-ordre sur leurs langages, semi-ordre que l'on définit avec les machines de Turing à semi-oracle. Cependant nous avons besoin d'une condition supplémentaire, le mélange uniforme, sur les décalages pour avoir équivalence des ordres. Ce résultat corrige le résultat énoncé dans [AS09], dont la démonstration manquait de précision quant à l'usage du semi-oracle.

Un pré-ordre sur les langages

Une *machine de Turing avec semi-oracle* \mathcal{L} , que l'on notera $\mathcal{M}^{\mathcal{L}}$ est une machine de Turing classique (voir la Définition 13) avec un ruban supplémentaire, le ruban d'oracle, qui contient initialement le mot vide et sur lequel la machine peut écrire au cours du calcul, ainsi qu'un état supplémentaire $q_?$, appelé l'état d'interrogation. Le langage \mathcal{L} est appelé le *langage oracle*. Le comportement de la machine est le même que celui d'une machine classique, avec en plus la règle suivante : lorsque l'état d'interrogation est atteint, soit le motif inscrit sur le ruban d'interrogation appartient au langage oracle, et dans ce cas la machine arrête son calcul, soit ce motif n'est pas dans le langage oracle et la machine continue son calcul.

Les règles de transition d'une machine à semi-oracle sont de la forme :

$$(Q \times \Gamma) \times (Q \times \Gamma \times Q \times \{\leftarrow, \cdot, \rightarrow\}).$$

Dans ce modèle on suppose donc que la machine ne possède qu'une tête de calcul sur le ruban de calcul, et une tête d'écriture seulement sur le ruban d'oracle, ces deux têtes ayant le même déplacement.

Par exemple, la règle $((q_1, y), (q_2, y', b' \leftarrow))$ correspond aux actions suivantes :

- la machine est dans l'état q_1 et lit la lettre y à la position de la tête de calcul son ruban de calcul ;

- la machine écrit la lettre y' sur son ruban de calcul et la lettre b' sur son ruban d'oracle, la tête de calcul passe dans l'état q_2 et se déplace vers la gauche, tout comme la tête d'écriture.

Comme dans le cas des machines de Turing classiques, il est possible de coder le comportement d'une machine à semi-oracle dans les règles locales d'un décalage de type fini. La règle précédemment citée sera par exemple représentée par le motif autorisé suivant

$$\begin{pmatrix} \left(q_2, \frac{x}{a} \right) & \frac{y'}{b'} & \frac{z}{c} \\ \frac{x}{a} & \left(q_1, \frac{y}{b} \right) & \frac{z}{c} \end{pmatrix},$$

où les lettres du ruban de calcul sont en vert et celles du ruban d'oracle en orange.

Exemple 3.2.1. On considère la machine avec semi-oracle $\mathcal{M}^{\mathcal{L}_{a^n b^n}}$ avec comme langage oracle $\mathcal{L}_{a^n b^n} = \{a^n b^n, n \in \mathbb{N}\}$, dont les règles de transition sont les suivantes

$$\begin{pmatrix} \left(q_b, \frac{x}{y} \right) & \frac{c}{\#} & \frac{z}{t} \\ \frac{x}{y} & \left(q_b, \frac{c}{\#} \right) & \frac{z}{t} \end{pmatrix}, \quad \begin{pmatrix} \frac{x}{y} & \left(q?, \frac{\#}{\#} \right) & \frac{z}{t} \\ \frac{x}{y} & \left(q\sim, \frac{\#}{\#} \right) & \frac{z}{t} \end{pmatrix},$$

mais aussi celles qui apparaissent dans l'exemple qui suit. Sur le mot d'entrée $p = aacbb$, la machine $\mathcal{M}^{\mathcal{L}_{a^n b^n}}$ fait le calcul suivant :

...	$\frac{\#}{\#}$	$\frac{a}{a}$	$\frac{a}{a}$	$\frac{c}{b}$	$\frac{B}{b}$	$\left(q_{stop}, \frac{B}{\#} \right)$	$\frac{\#}{\#}$	$\frac{\#}{\#}$...
...	$\frac{\#}{\#}$	$\frac{a}{a}$	$\frac{a}{a}$	$\frac{c}{b}$	$\frac{B}{b}$	$\left(q?, \frac{B}{\#} \right)$	$\frac{\#}{\#}$	$\frac{\#}{\#}$...
...	$\frac{\#}{\#}$	$\frac{a}{a}$	$\frac{a}{a}$	$\frac{c}{b}$	$\frac{B}{b}$	$\left(q\sim, \frac{B}{0} \right)$	$\frac{\#}{\#}$	$\frac{\#}{\#}$...
...	$\frac{\#}{\#}$	$\frac{a}{a}$	$\frac{a}{a}$	$\frac{c}{b}$	$\frac{B}{b}$	$\frac{B}{0}$	$\left(q_0, \frac{\#}{\#} \right)$	$\frac{\#}{\#}$...
...	$\frac{\#}{\#}$	$\frac{a}{a}$	$\frac{a}{a}$	$\frac{c}{b}$	$\frac{B}{b}$	$\left(q_0, \frac{B}{\#} \right)$	$\frac{\#}{\#}$	$\frac{\#}{\#}$...
...	$\frac{\#}{\#}$	$\frac{a}{a}$	$\frac{a}{a}$	$\frac{c}{b}$	$\left(q_b, \frac{B}{0} \right)$	$\frac{B}{\#}$	$\frac{\#}{\#}$	$\frac{\#}{\#}$...
...	$\frac{\#}{\#}$	$\frac{a}{a}$	$\frac{a}{a}$	$\frac{c}{b}$	$\frac{B}{0}$	$\left(q_c, \frac{b}{\#} \right)$	$\frac{\#}{\#}$	$\frac{\#}{\#}$...
...	$\frac{\#}{\#}$	$\frac{a}{a}$	$\frac{a}{a}$	$\frac{c}{b}$	$\left(q_0, \frac{B}{\#} \right)$	$\frac{b}{\#}$	$\frac{\#}{\#}$	$\frac{\#}{\#}$...
...	$\frac{\#}{\#}$	$\frac{a}{a}$	$\frac{a}{a}$	$\left(q_b, \frac{c}{0} \right)$	$\frac{B}{\#}$	$\frac{b}{\#}$	$\frac{\#}{\#}$	$\frac{\#}{\#}$...
...	$\frac{\#}{\#}$	$\frac{a}{a}$	$\frac{a}{a}$	$\frac{c}{0}$	$\left(q_c, \frac{b}{\#} \right)$	$\frac{b}{\#}$	$\frac{\#}{\#}$	$\frac{\#}{\#}$...
...	$\frac{\#}{\#}$	$\frac{a}{a}$	$\frac{a}{a}$	$\left(q_0, \frac{c}{\#} \right)$	$\frac{b}{\#}$	$\frac{b}{\#}$	$\frac{\#}{\#}$	$\frac{\#}{\#}$...
...	$\frac{\#}{\#}$	$\frac{a}{a}$	$\left(q_0, \frac{a}{\#} \right)$	$\frac{c}{\#}$	$\frac{b}{\#}$	$\frac{b}{\#}$	$\frac{\#}{\#}$	$\frac{\#}{\#}$...
...	$\frac{\#}{\#}$	$\left(q_0, \frac{a}{\#} \right)$	$\frac{a}{\#}$	$\frac{c}{\#}$	$\frac{b}{\#}$	$\frac{b}{\#}$	$\frac{\#}{\#}$	$\frac{\#}{\#}$...

Cette machine $\mathcal{M}^{\mathcal{L}_{a^n b^n}}$ reconnaît donc le langage $\mathcal{L}_{\mathcal{M}} = \{a^n c^* b^n, n \in \mathbb{N}\}$

Ces machines de Turing à semi-oracle nous permettent de définir un pré-ordre sur les langages. On dit qu'un langage \mathcal{L} est plus petit qu'un langage \mathcal{L}' , et l'on notera $\mathcal{L} \preceq \mathcal{L}'$, s'il existe une machine de Turing $\mathcal{M}^{\mathcal{L}'}$ avec semi-oracle \mathcal{L}' qui s'arrête exactement sur les mots du langage \mathcal{L} , autrement dit $\text{dom}(\mathcal{M}^{\mathcal{L}'}) = \mathcal{L}$.

Proposition 3.2.1. *La relation \preceq est un pré-ordre.*

Démonstration. Il suffit de vérifier que la relation \preceq est réflexive et transitive. La réflexivité s'obtient en considérant, pour chaque langage \mathcal{L} , la machine de Turing avec oracle \mathcal{L} qui fait directement appel à cet oracle sur le mot d'entrée. Ainsi on a bien $\mathcal{L} \preceq \mathcal{L}$ pour tout langage.

Pour la transitivité, supposons que l'on ait $\mathcal{L}_1 \preceq \mathcal{L}_2$ et $\mathcal{L}_2 \preceq \mathcal{L}_3$. Il existe donc une machine \mathcal{M}_2 avec semi-oracle \mathcal{L}_2 telle que $\text{dom}(\mathcal{M}_2) = \mathcal{L}_1$, et une machine \mathcal{M}_3 avec semi-oracle \mathcal{L}_3 telle que $\text{dom}(\mathcal{M}_3) = \mathcal{L}_2$. On construit une machine \mathcal{M} qui, sur un mot d'entrée p , simule le comportement de \mathcal{M}_2 , et dès que \mathcal{M}_2 fait appel à son oracle \mathcal{L}_2 , la machine simule le comportement de \mathcal{M}_3 . Ainsi \mathcal{M} s'arrête sur le mot p si et seulement si $p \in \mathcal{L}_1$, et cette machine a \mathcal{L}_3 comme semi-oracle. On a donc bien $\mathcal{L}_1 \preceq \mathcal{L}_3$. \square

Nous définissons aussi la relation d'équivalence associée à ce pré-ordre. On notera $\mathcal{L} \approx \mathcal{L}'$ si et seulement si $\mathcal{L} \preceq \mathcal{L}'$ et $\mathcal{L}' \preceq \mathcal{L}$. Cette relation d'équivalence définit des classes de langages que l'on peut comparer par le biais du semi-ordre. Par exemple, la classe des langages récursivement énumérables est la plus petite pour le semi-ordre \preceq , et on a $\emptyset \approx \mathcal{L}$ pour tout langage \mathcal{L} récursivement énumérable.

Théorème de clôture

Remarquons tout d'abord que pour tout décalage effectif \mathbf{T}'' , il est possible de construire n'importe quel décalage de type fini (et ce quelle que soit sa dimension) simplement à l'aide des opérations **Prod** (pour définir le bon alphabet), **TF** (pour ajouter les règles du décalage de type fini) et **SE** (pour se placer dans la bonne dimension). En terme de clôture, cela s'écrit

$$Cl_{\mathbf{Prod}, \mathbf{Fact}, \mathbf{TF}, \mathbf{SA}, \mathbf{SE}}(\mathbf{T}'') \supset \mathcal{SFT}.$$

Le Théorème 3.1.1 nous apprend que pour tout décalage effectif \mathbf{T}' de dimension d , il est possible de construire un décalage de type fini de dimension $d + 1$ qui simule \mathbf{T}' par les opérations **Fact** et **SA**. Ainsi $\mathbf{T}' \leq_{\mathbf{Prod}, \mathbf{Fact}, \mathbf{TF}, \mathbf{SA}, \mathbf{SE}} \mathbf{T}''$. En particulier cela signifie que tous les décalages effectifs appartiennent à la même classe pour le pré-ordre $\leq_{\mathbf{Prod}, \mathbf{Fact}, \mathbf{TF}, \mathbf{SA}, \mathbf{SE}}$. Nous avons exactement la même situation pour les langages récursivement énumérables et le pré-ordre \preceq .

Nous allons voir qu'il est possible d'adapter la preuve du Théorème 3.1.1 pour en obtenir une généralisation, généralisation qui s'exprime sous forme d'une correspondance entre le pré-ordre sur les langages \preceq et le pré-ordre sur les décalages $\leq_{\text{Prod,Fact,TF,SA,SE}}$. Malheureusement cette généralisation n'est que partielle puisqu'elle nécessite une condition supplémentaire sur les décalages.

Définition 14. Un décalage $\Sigma \subset A^{\mathbb{Z}^d}$ est dit *uniformément mélangeant* s'il existe un entier $N \in \mathbb{N}$ tel que pour tout $u \in \mathcal{L}_m(\Sigma)$, $v \in \mathcal{L}_{m'}(\Sigma)$ et pour tout $n \in \mathbb{Z}^d$ qui vérifie $|n| \geq N$, il existe $x \in \Sigma$ tel que $x|_{\mathbb{S}_m} = u$ et $\sigma^n(x)|_{\mathbb{S}_{m'}} = v$.

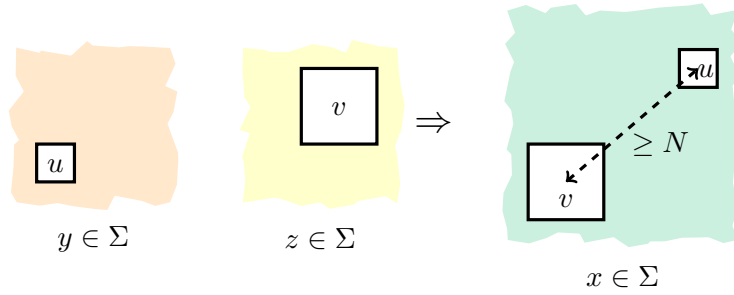


FIGURE 25: Deux motifs du langage d'un décalage uniformément mélangeant apparaissent dans une même configuration, à condition qu'ils soient à une certaine distance l'un de l'autre.

En d'autres termes, deux motifs apparaissant dans un décalage \mathbf{T} uniformément mélangeant apparaissent aussi dans une même configuration du décalage, à condition qu'ils soient à une certaine distance l'un de l'autre (voir la Figure 25). Sous cette condition sur un décalage \mathbf{T} , il est possible de caractériser l'ensemble des décalages simulés par \mathbf{T} à l'aide des cinq opérations définies précédemment.

Théorème 3.2.2. Soit \mathbf{T} un décalage uniformément mélangeant. Alors :

$$\text{Cl}_{\text{Prod,Fact,TF,SA,SE}}(\mathbf{T}) = \{\mathbf{T}_{\mathcal{L}} : \mathcal{L} \preceq \mathcal{L}(\mathbf{T})^c\}.$$

De manière équivalente, si \mathbf{T}' et \mathbf{T}'' sont deux décalages, et que \mathbf{T}'' est uniformément mélangeant, on a :

$$\mathbf{T}' \leq_{\text{Prod,Fact,TF,SA,SE}} \mathbf{T}'' \iff \mathcal{L}(\mathbf{T}')^c \preceq \mathcal{L}(\mathbf{T}'')^c.$$

Démonstration. Nous démontrons la première formulation du résultat par double inclusion. Il est à noter que l'inclusion directe reste vraie sans l'hypothèse de mélange uniforme. Dans cette démonstration, la notation \mathcal{P}_A^d désigne l'ensemble des motifs rectangulaires d -dimensionnels sur l'alphabet A .

Inclusion directe On note $\mathcal{L} = \mathcal{L}(\mathbf{T})^c$. Pour avoir la relation

$$\mathcal{C}l_{\mathbf{Prod}, \mathbf{Fact}, \mathbf{TF}, \mathbf{SA}, \mathbf{SE}}(\mathbf{T}) \subseteq \{\mathbf{T}_{\mathcal{L}'} : \mathcal{L}' \preceq \mathcal{L}\}$$

il suffit de montrer que que l'ensemble $\{\mathbf{T}_{\mathcal{L}'} : \mathcal{L}' \preceq \mathcal{L}\}$ est stable pour chacune des cinq opérations. Soient $\mathcal{L}_1 \subseteq \mathcal{P}_{A_1}^{d_1}$ et $\mathcal{L}_2 \subseteq \mathcal{P}_{A_2}^{d_2}$ deux langages tels que $\mathcal{L}_i \preceq \mathcal{L}$ pour $i \in \{1, 2\}$. Alors pour chaque $i \in \{1, 2\}$, il existe une machine de Turing \mathcal{M}_i avec semi-oracle \mathcal{L} dont le domaine est exactement \mathcal{L}_i .

- **Stabilité par produit Prod :** Soit $\mathbf{T}' = \mathbf{Prod}(\mathbf{T}_1) \mathbf{T}_2$. Alors $\mathbf{T}' = \mathbf{T}_{\mathcal{L}'}$ avec $\mathcal{L}' = \mathcal{L}_1 \times \mathcal{P}_{A_2}^{d_2} \cup \mathcal{P}_{A_1}^{d_1} \times \mathcal{L}_2$, car un motif produit est interdit dans le décalage produit si et seulement si l'un des motifs qui le compose est interdit dans le décalage associé. Le langage \mathcal{L}' peut être vu comme le domaine d'une machine de Turing \mathcal{M}' avec semi-oracle \mathcal{L} . Il suffit pour cela de simuler les deux machines \mathcal{M}_1 et \mathcal{M}_2 (chaque machine effectue une étape de calcul à tour de rôle) sur chacune des coordonnées d'un motif de \mathcal{L}' . Ainsi $\mathcal{L}' \preceq \mathcal{L}$.

- **Stabilité par opération type fini TF :** Soit $\mathbf{T}' = \mathbf{TF}_P(\mathbf{T}_{\mathcal{L}_1})$. Comme P est fini, nous avons $\mathcal{L}_1 \cup P \preceq \mathcal{L}_1 \preceq \mathcal{L}$ et $\mathbf{T}' = \mathbf{T}_{\mathcal{L}_1 \cup P}$.

- **Stabilité par facteur Fact :** Soit $\mathbf{T}' = \mathbf{Fact}_{\Pi}(\mathbf{T}_{\mathcal{L}_1})$ où $\Pi : A_1^{\mathbb{Z}^{d_1}} \rightarrow B^{\mathbb{Z}^{d_1}}$ est une fonction n -bloc dont la fonction locale est π . Alors $\mathbf{T}' = \mathbf{T}_{\mathcal{L}'}$ avec $\mathcal{L}' = (\pi(\mathcal{L}_1^c))^c$. De plus $\mathcal{L}' \preceq \mathcal{L}_1$. En effet si $p \in \mathcal{P}_B^{d_1}$, il suffit de simuler la machine \mathcal{M}_1 sur tous les motifs $p' \in A^{supp(p) + \mathbb{S}_n^{d_1}}$ tels que $\bar{\pi}(p') = p$, tous les calculs étant lancés les uns à la suite des autres en parallèle (la machine consacre à tour de rôle une étape de calcul à chacun des motifs).

- **Stabilité par sous-action projective SA :** Soit $\mathbf{T}' = \mathbf{SA}_{\mathbb{G}}(\mathbf{T}_{\mathcal{L}_1}) \subseteq A_1^{\mathbb{Z}^{d'}}$ avec \mathbb{G} un sous-groupe de \mathbb{Z}^{d_1} de dimension $d' \leq d_1$. Considérons le langage $\mathcal{L}' \subseteq \mathcal{P}_{A_1}^{d_1}$ qui est le domaine de la machine de Turing \mathcal{M}' suivante : sur un motif $p \in \mathcal{P}_{A_1}^{d'}$ de support \mathbb{U} , la machine \mathcal{M}' simule à tour de rôle la machine \mathcal{M}_1 sur chacun des motifs de support $[-n; n]^{d_1}$ qui complète le motif p dans $\mathcal{P}_{A_1}^{d_1}$, où $[-n; n]^{d_1}$ est le support minimal contenant la projection de \mathbb{U} dans \mathbb{G} . Ainsi $\mathcal{L}' \preceq \mathcal{L}_1$, et $\mathbf{T}' = \mathbf{T}_{\mathcal{L}'}$.

- **Stabilité par extension SE :** Soit $\mathbf{T}' = \mathbf{SE}_{\mathbb{G}, \mathbb{G}'}(\mathbf{T}_{\mathcal{L}_1})$ avec \mathbb{G} isomorphe à \mathbb{Z}^{d_1} et $\mathbb{G} \oplus \mathbb{G}' = \mathbb{Z}^{d_1+d}$. Soit $\mathcal{L}' \subseteq \mathcal{P}_{A_1}^{d_1+d}$ le langage dont chaque motif p est la superposition de motifs $p_1, \dots, p_d \in \mathcal{P}_{A_1}^{d_1}$ de sorte qu'il existe $i \in \{1, \dots, d\}$ tel que $p_i \in \mathcal{L}_1$. Alors $\mathcal{L}' \preceq \mathcal{L}_1$ et $\mathbf{T}' = \mathbf{T}_{\mathcal{L}'}$.

Inclusion réciproque Soient \mathbf{T}'' un décalage uniformément mélangeant de dimension d'' et \mathbf{T}' un décalage de dimension d' tel que $\mathcal{L}(\mathbf{T}')^c \preceq \mathcal{L}(\mathbf{T}'')^c$. Nous construisons un décalage $\mathbf{T}_{\text{Oracle}}$ à l'aide du décalage \mathbf{T}'' et des opérations **Prod**, **TF** et **SE** simulant le décalage \mathbf{T}' par les opérations **Fact** et **SA**. La construction est une adaptation de la construction présentée dans la Partie 3.1, et nous nous restreignons au cas où $d'' = d' = 1$. Il suffit de remplacer l'usage des machines de Turing classique par des machines de Turing avec semi-oracle. Pour cela nous avons besoin d'une dimension supplémentaire, appelée *dimension oracle*, pour coder l'oracle, de sorte que si les décalages \mathbf{T}'' et \mathbf{T}' sont de dimension 1, la construction est de dimension 3. Nous décrivons ici les changements par rapport à la construction du décalage \mathbf{T} de la Partie 3.1.5. Le premier est l'ajout d'un cinquième niveau, que nous allons décrire plus bas. Les autres changements sont les suivants :

1. La machine $\mathcal{M}_{\text{Forbid}}$ dispose d'un ruban supplémentaire, le ruban d'oracle, sur lequel elle écrit un mot w au cours de ses calculs. Lorsque cette machine $\mathcal{M}_{\text{Forbid}}$ atteint l'état d'interrogation $q_?$, elle compare le mot w inscrit sur le ruban d'oracle avec le mot de $\mathcal{L}(\mathbf{T}'')$ qui apparaît au début de sa zone de calcul, dans le décalage du cinquième niveau. Pour une raison technique qui sera explicitée juste après, la machine $\mathcal{M}_{\text{Forbid}}$ interrompt son calcul pendant $M + |w|$ étapes de calcul, où M est la constante de mélange. Pendant ces $M + |w|$ étapes de calcul, la machine $\mathcal{M}_{\text{Forbid}}$ effectue la comparaison. Une fois ce temps écoulé, la machine $\mathcal{M}_{\text{Forbid}}$ continue son calcul si le mot w est dans $\mathcal{L}(\mathbf{T}'')$, ce qui signifie que le mot w n'est pas un motif interdit de \mathbf{T}'' . Sinon elle arrête son calcul et entre dans un état spécial q_{stop} dont la présence est interdite dans le décalage final.
2. Par des opérations **SE** et **TF** nous superposons des configurations de $\mathbf{T}_{\text{Final}}$ de façon à synchroniser les zones de calcul, mais les calculs à l'intérieur de ces zones restent indépendants les uns des autres :

$$\mathbf{T}_{\text{SyncroZone}} = \mathbf{TF}_{\text{SyncroZone}} \left(\mathbf{SE}_{\mathbb{Z}^2, \mathbb{Z}} (\mathbf{T}_{\text{Final}}) \right).$$

3. Par une opération **SE**, nous remplissons l'espace avec des configurations de \mathbf{T}'' le long de la dimension oracle, et ces configurations sont indépendantes. Puis nous ajoutons l'oracle à cette construction de façon à former un cinquième niveau :

$$\mathbf{T}_{\text{Niveau 5}} = \mathbf{Prod} \left(\mathbf{T}_{\text{SyncroZone}}, \mathbf{SE}_{\mathbb{Z}, \mathbb{Z}^2} (\mathbf{T}'') \right).$$

4. Dans chaque zone de calcul, l'oracle doit être le même. De plus pour le comparer avec le mot w inscrit sur le ruban d'oracle, l'oracle est décalé à chaque tuile de calcul rencontrée. Tout ceci est possible grâce à la synchronisation par la condition **SynchroZone**, et est réalisé par des conditions de type fini **Shift** (voir la Figure 26).
5. Lorsqu'une machine $\mathcal{M}_{\text{Forbid}}$ atteint son état d'interrogation $q?$, la comparaison entre le mot sur le ruban d'oracle et le mot de la configuration oracle s'effectue en une seule étape de calcul. Si le mot inscrit sur le ruban d'oracle ne correspond pas, la configuration est rejetée. Une fois cette comparaison effectuée, les machines $\mathcal{M}_{\text{Forbid}}$ situées au dessus et en dessous (selon la direction oracle) de la première machine ne peuvent pas utiliser la configuration oracle, et doivent attendre leur tour. Cette attente dure $M + |w|$ étapes de calcul, où M est la constante de mélange, et où la longueur $|w|$ est bornée par la taille du ruban de la zone de calcul utilisée. La machine peut facilement calculer cette quantité, il lui suffit de parcourir son ruban d'oracle jusqu'à la fin du mot w (ceci prend $|w|$ étapes de calcul), et de compter jusqu'à M qui est une constante que l'on code directement dans la machine. Toutes ces contraintes sont codées par des conditions de type fini **Questioning**. Nous obtenons le décalage :

$$\mathbf{T}_{\text{Oracle}} = \mathbf{TF}^{\text{Shift} \cup \text{Questioning}}(\mathbf{T}_{\text{Niveau 5}}).$$

Avec toutes ces modifications, le décalage $\mathbf{T}_{\text{Oracle}}$ est tel que :

$$\mathbf{T}' = \mathbf{Fact}_{\pi}(\mathbf{SA}_{(1,0,0)\mathbb{Z}}(\mathbf{T}_{\text{Oracle}})),$$

où est π est le facteur qui consiste à ne garder que les symboles de l'alphabet de \mathbf{T}' (la preuve est similaire à celle de la Partie 3.1.5). Ceci achève la démonstration. \square

Ainsi le Théorème 3.2.2 n'est valable que pour des décalages uniformément mélangeants. La proposition suivante montre que dans une classe d'équivalence de langages pour le semi-ordre \approx , il est toujours possible de trouver un langage de motifs interdits qui définit un décalage uniformément mélangeant.

Proposition 3.2.3. *Pour tout décalage \mathbf{T} , il existe un langage \mathcal{L}' équivalent au complémentaire du langage $\mathcal{L}(\mathbf{T})^c$ et tel que le décalage $\mathbf{T}_{\mathcal{L}'}$ est uniformément mélangeant.*

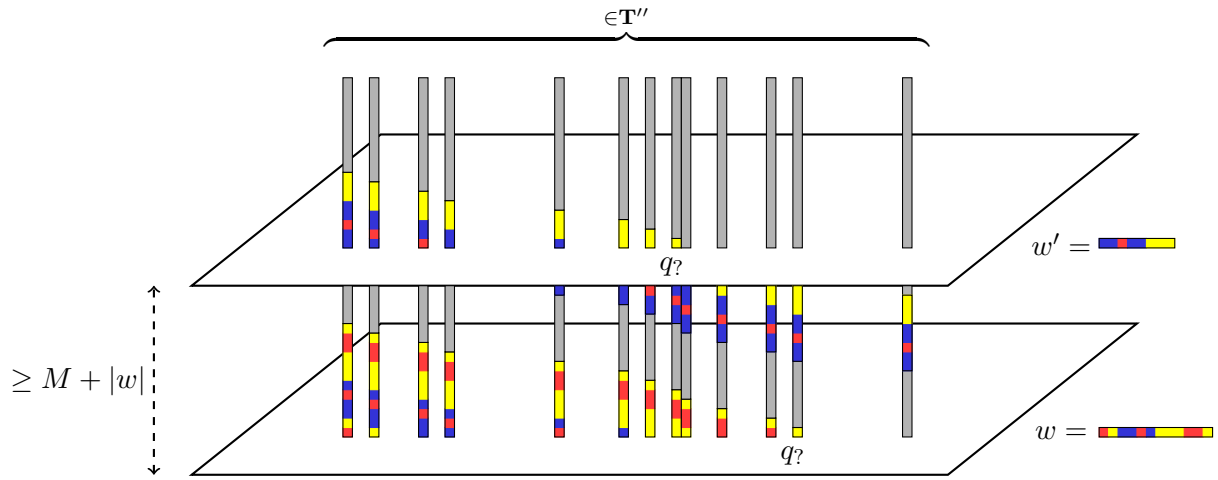


FIGURE 26: Une illustration de la construction de dimension 3, dans laquelle la dimension oracle est verticale. Les zones de calcul considérées sont des zones de niveau 4. Les deux machines $\mathcal{M}_{\text{Forbid}}$ représentées peuvent utiliser une même configuration oracle, à condition que chacune d'entre elle attende suffisamment ($M + |w|$ étapes de calcul suffisent) pour être sûre que la partie de la configuration oracle qu'elle utilise ne dépend pas de l'usage qu'en ferait une autre machine.

Démonstration. Soit \mathbf{T} un décalage sur l'alphabet $A = \{a_1, \dots, a_n\}$. On définit un langage \mathcal{L}_0 sur l'alphabet $A' = A \cup \{\#\}$ de la manière suivante :

$$\mathcal{L}_0 = \text{Suff}(\mathcal{L}(\mathbf{T})).\{\#\}. (\mathcal{L}(\mathbf{T})^+.\{\#\})^* .\text{Pref}(\mathcal{L}(\mathbf{T})),$$

où $\text{Suff}(\mathcal{L})$ est l'ensemble des suffixes des mots de \mathcal{L} , $\text{Pref}(\mathcal{L})$ est l'ensemble des préfixes de mots de \mathcal{L} et \mathcal{L}^+ est l'ensemble $\mathcal{L} \setminus \{\varepsilon\}$ où ε est le mot vide. Les mots de ce langage sont donc des concaténations de mots de $\mathcal{L}(\mathbf{T})$, deux mots étant séparés par un symbole $\#$. Les premiers et derniers mots de ces concaténations sont éventuellement tronqués, d'où la présence dans la définition de \mathcal{L}_0 des ensembles $\text{Suff}(\mathcal{L})$ et $\text{Pref}(\mathcal{L})$.

On appelle \mathcal{L}' le complémentaire du langage \mathcal{L}_0 . Remarquons tout d'abord que $\mathcal{L}' \approx \mathcal{L}(\mathbf{T})^c$, tout simplement car $\mathcal{L}(\mathbf{T})^c \preceq \mathcal{L}'$. Réciproquement, on construit une machine de Turing avec semi-oracle $\mathcal{L}(\mathbf{T})^c$ qui termine exactement sur

les mots du langage \mathcal{L}' . Sur un mot d'entrée w , cette machine commence par détecter l'éventuelle présence de deux symboles $\#$ consécutifs. Si la machine en trouve, elle s'arrête. Sinon le mot w peut être décomposé de la manière suivante :

$$w = u_1\#u_2\#\dots\#u_{k-1}\#u_k$$

avec $u_i \in A'^*$ et de sorte que les mots u_2, \dots, u_{k-1} ne sont pas le mot vide ε . Pour tout entier $2 \leq i \leq k-1$, la machine fait appel à l'oracle pour savoir si $u_i \in \mathcal{L}(\mathbf{T})^c$. S'il existe un entier i tel que $u_i \in \mathcal{L}(\mathbf{T})^c$, alors $u_i \notin \mathcal{L}(\mathbf{T})$ et donc $w \notin \mathcal{L}_0$, d'où $w \in \mathcal{L}'$. Si la machine ne s'est pas arrêtée après avoir testé tous les mots u_2, \dots, u_{k-1} , il reste à vérifier que $u_1 \notin \text{Suff}(\mathcal{L})$ et que $u_k \notin \text{Pref}(\mathcal{L})$. Pour ce faire, la machine interroge l'oracle avec les mots $a_1u_1, \dots, a_nu_1, a_1a_1u_1, a_1a_2u_1, \dots$ et $u_ka_1, \dots, u_ka_n, u_ka_1a_1, u_ka_1a_2, \dots$. De cette façon, si $w \in \mathcal{L}'$ la machine finira par s'arrêter, et donc $\mathcal{L}' \preceq \mathcal{L}(\mathbf{T})^c$.

Le langage \mathcal{L}_0 est à la fois factoriel et extensible, c'est donc le langage d'un décalage \mathbf{T}_0 que l'on peut décrire facilement :

$$\mathbf{T}_0 = \left\{ x \in A'^{\mathbb{Z}} \mid x = \dots\#u_{-n}\#\dots\#u_{-1}\#u_0\#u_1\#\dots\#u_n\#\dots \text{ avec } u_i \in \mathcal{L}(X) \setminus \epsilon \right\}.$$

Le décalage \mathbf{T}_0 est uniformément mélangeant avec $M = 1$ comme constante de mélange, ce qui signifie que $\mathbf{T}_{\mathcal{L}'}$ est uniformément mélangeant et achève la démonstration. □

De la Proposition 3.2.3 et du Théorème 3.2.2 nous déduisons le corollaire suivant :

Corollaire 3.2.4. *Soit \mathbf{T} un décalage et $\mathcal{L} = \mathcal{L}(\mathbf{T})^c$ le complémentaire de son langage. Alors*

$$Cl_{\text{Prod,Fact,TF,SA,SE}}(\{\mathbf{T}_{\mathcal{L}'} : \mathcal{L}' \preceq \mathcal{L}\}) = \{\mathbf{T}_{\mathcal{L}'} : \mathcal{L}' \preceq \mathcal{L}\}.$$

Nous obtenons ainsi une manière de construire des classes de décalages stables pour les cinq opérations. Cependant la question de savoir caractériser toutes ces classes reste ouverte.

3.2.2 Décalages S-adiques multidimensionnels effectifs

Dans cette partie nous nous intéressons aux décalage S-adiques multidimensionnels et cherchons à les situer parmi les trois classes présentées dans la Partie 1.3. Cette classe ne peut pas être incluse dans la classe des décalages sofiques par

une simple raison de cardinalité : il existe un nombre dénombrables de décalages sofiques, tandis que choisir une suite infinie de substitutions parmi un ensemble fini \mathcal{S} peut se faire de manière indénombrable. Nous montrons dans cette partie que les décalages S-adiques multidimensionnels qui sont sofiques sont exactement ceux dont la suite S qui les définit est effective.

Pour montrer ce résultat, nous utilisons celui de Mozes [Moz89], qui montre qu'un décalage substitutif $\mathbf{T}_{\mathcal{S}}$ est sofique. A chaque niveau d'itération de cette preuve, plusieurs substitutions de \mathcal{S} sont en général utilisées. Le but de notre construction est donc de réussir à synchroniser ces substitutions, et pour ce faire nous utilisons un décalage effectif de dimension 1 qui code la suite S de substitutions. Ce décalage effectif peut se retrouver dans un décalage sofique de dimension 3 [Hoc09], ou même de dimension 2 (voir le Théorème 3.1.1 ou bien [DRS10b]).

Les décalages S-adiques multidimensionnels effectifs sont sofiques

Soit $S \in \mathcal{S}^{\mathbb{N}}$ une suite de substitutions choisies parmi un ensemble fini \mathcal{S} . L'inclusion $\mathbf{T}_S \subset \mathbf{T}_{\mathcal{S}}$ est toujours vérifiée. Par le théorème de Mozes [Moz89], on sait que le décalage $\mathbf{T}_{\mathcal{S}}$ est sofique si l'ensemble \mathcal{S} vérifie la propriété A, mais il n'y a aucun raison évidente pour que \mathbf{T}_S le soit aussi.

Théorème 3.2.5. *Soit \mathcal{S} un ensemble fini de substitutions multidimensionnelles non dégénérées et $S \in \mathcal{S}^{\mathbb{N}}$ une suite effective de substitutions. Alors le décalage \mathbf{T}'_S est sofique. Si de plus l'ensemble \mathcal{S} possède la propriété A, alors le décalage \mathbf{T}_S est lui aussi sofique.*

Remarque. Nous présentons la preuve du Théorème 3.2.5 seulement pour \mathbf{T}_S . La démonstration est similaire pour \mathbf{T}'_S , puisqu'il suffit de remplacer le décalage $\mathbf{T}_{\mathcal{S}}$ par $\mathbf{T}'_{\mathcal{S}}$ dans la construction.

Démonstration. Supposons que $d = 2$, la démonstration étant similaire lorsque $d \geq 3$. Soit \mathcal{S} un ensemble fini de $(A, 2)$ -substitutions non dégénérées. Nous définissons un nouvel alphabet $A' = A \times \mathcal{S}^2$. A chaque substitution $s \in \mathcal{S}$ nous associons une $(A', 2)$ -substitution \tilde{s} de même support, définie par

$$\tilde{s} : (a, s_V, s_H) \mapsto \begin{array}{c|ccc|c} \begin{array}{c} (s(a)_{(0, \mathbf{k}_2^s(a))}, s, s_H) \\ (s(a)_{(0, \mathbf{k}_2^s(a)-1)}, s, s) \\ \vdots \\ (s(a)_{(0,1)}, s, s) \\ (s(a)_{(0,0)}, s, s) \end{array} & \begin{array}{c} (s(a)_{(1, \mathbf{k}_2^s(a))}, s, s_H) \\ \dots \\ (s(a)_{(i,j)}, s, s) \\ \dots \\ (s(a)_{(1,0)}, s, s) \end{array} & \dots & \begin{array}{c} (s(a)_{(\mathbf{k}_1^s(a)-1, \mathbf{k}_2^s(a))}, s, s_H) \\ \dots \\ (s(a)_{(\mathbf{k}_1^s(a)-1,0)}, s, s) \end{array} & \begin{array}{c} (s(a)_{(\mathbf{k}_1^s(a), \mathbf{k}_2^s(a))}, s_V, s_H) \\ (s(a)_{(\mathbf{k}_1^s(a), \mathbf{k}_2^s(a)-1)}, s_V, s) \\ \vdots \\ (s(a)_{(\mathbf{k}_1^s(a), 1)}, s_V, s) \\ (s(a)_{(\mathbf{k}_1^s(a), 0)}, s_V, s) \end{array} \end{array}$$

Toutes ces substitutions \tilde{s} forment un ensemble fini $\tilde{\mathcal{S}} = \{\tilde{s} : s \in \mathcal{S}\}$. Soit $S = (s_i)_{i \in \mathbb{N}} \in \mathcal{S}^{\mathbb{N}}$ une suite effective, nous pouvons donc lui associer la suite

effective $\tilde{S} = (\tilde{s}_i)_{i \in \mathbb{N}} \in \tilde{\mathcal{S}}^{\mathbb{N}}$. L'objectif de ces substitutions \tilde{s} est de garder en mémoire la suite des substitutions précédemment appliquées. Pour cela une substitution s appliquée à un certain niveau d'itération est gardée en mémoire et transférée aux niveaux suivants grâce aux substitutions s_V (pour transfert vertical) et s_H (pour transfert horizontal) de l'alphabet A' . Ces transferts sont visibles sur les dernières ligne et colonne d'un motif $\tilde{s}(a, s_V, s_H)$.

Exemple 3.2.2. Soit \mathcal{S} l'ensemble des substitutions sur l'alphabet $A = \{\circ, \bullet\}$ définies dans l'Exemple 2.1.3. Soit la suite $\tilde{S} = (\tilde{s}_d, \tilde{s}_d, \tilde{s}_a, \tilde{s}_a, \dots)$. Appliquée à la lettre \bullet de l'alphabet, nous obtenons les motifs présentés ci-dessous.

$$(\bullet, s_3, s_3) \xrightarrow{\tilde{s}_2} \begin{array}{cc} (\circ, s_2, s_3) & (\circ, s_3, s_3) \\ (\bullet, s_2, s_2) & (\circ, s_3, s_2) \end{array} \xrightarrow{\tilde{s}_1} \begin{array}{cc|cc} (\circ, s_1, s_3) & (\circ, s_2, s_3) & (\circ, s_1, s_3) & (\circ, s_3, s_3) \\ (\circ, s_1, s_1) & (\circ, s_2, s_1) & (\circ, s_1, s_1) & (\circ, s_3, s_1) \\ \hline (\circ, s_1, s_2) & (\circ, s_2, s_2) & (\circ, s_1, s_2) & (\circ, s_3, s_2) \\ (\bullet, s_1, s_1) & (\circ, s_2, s_1) & (\circ, s_1, s_1) & (\circ, s_3, s_1) \end{array}$$

(\bullet, s_0, s_3)	(\bullet, s_0, s_3)	(\bullet, s_1, s_3)	(\bullet, s_0, s_3)	(\bullet, s_0, s_3)	(\bullet, s_2, s_3)	(\bullet, s_0, s_3)	(\bullet, s_0, s_3)	(\bullet, s_1, s_3)	(\bullet, s_0, s_3)	(\bullet, s_0, s_3)	(\bullet, s_3, s_3)
(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_1, s_0)	(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_2, s_0)	(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_1, s_0)	(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_3, s_0)
(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_1, s_0)	(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_2, s_0)	(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_1, s_0)	(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_3, s_0)
(\bullet, s_0, s_1)	(\bullet, s_0, s_1)	(\bullet, s_1, s_1)	(\bullet, s_0, s_1)	(\bullet, s_0, s_1)	(\bullet, s_2, s_1)	(\bullet, s_0, s_1)	(\bullet, s_0, s_1)	(\bullet, s_1, s_1)	(\bullet, s_0, s_1)	(\bullet, s_0, s_1)	(\bullet, s_3, s_1)
(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_1, s_0)	(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_2, s_0)	(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_1, s_0)	(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_3, s_0)
(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_1, s_0)	(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_2, s_0)	(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_1, s_0)	(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_3, s_0)
(\bullet, s_0, s_2)	(\bullet, s_0, s_2)	(\bullet, s_1, s_2)	(\bullet, s_0, s_2)	(\bullet, s_0, s_2)	(\bullet, s_2, s_2)	(\bullet, s_0, s_2)	(\bullet, s_0, s_2)	(\bullet, s_1, s_2)	(\bullet, s_0, s_2)	(\bullet, s_0, s_2)	(\bullet, s_3, s_2)
(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_1, s_0)	(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_2, s_0)	(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_1, s_0)	(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_3, s_0)
(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_1, s_0)	(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_2, s_0)	(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_1, s_0)	(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_3, s_0)
(\circ, s_0, s_1)	(\circ, s_0, s_1)	(\circ, s_1, s_1)	(\bullet, s_0, s_1)	(\bullet, s_0, s_1)	(\bullet, s_2, s_1)	(\bullet, s_0, s_1)	(\bullet, s_0, s_1)	(\bullet, s_1, s_1)	(\bullet, s_0, s_1)	(\bullet, s_0, s_1)	(\bullet, s_3, s_1)
(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_1, s_0)	(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_2, s_0)	(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_1, s_0)	(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_3, s_0)
(\bullet, s_0, s_0)	(\bullet, s_0, s_0)	(\bullet, s_1, s_0)	(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_2, s_0)	(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_1, s_0)	(\circ, s_0, s_0)	(\circ, s_0, s_0)	(\circ, s_3, s_0)

Sur la ligne inférieure du dernier motif, on voit apparaître la suite de substitutions $s_0 \circ s_1 \circ s_2$ qui a permis de le définir.

Considérons la fonction 1-bloc $\pi : A' \rightarrow A$ qui ne conserve que la lettre de A et $\pi_V : A' \rightarrow \mathcal{S}$ (resp. $\pi_H : A' \rightarrow \mathcal{S}$) la fonction 1-bloc qui ne conserve que la substitution $s_V \in \mathcal{S}$ (resp. $s_H \in \mathcal{S}$) d'un élément $(a, s_V, s_H) \in A'$.

Proposition 3.2.6. $\mathbf{T}_S = \pi(\mathbf{T}_{\tilde{S}})$

Démonstration. Ce résultat provient directement du fait que l'alphabet A' contient l'alphabet A , et que la substitution \tilde{s} restreinte à l'alphabet A est exactement la substitution s . \square

Il nous suffit donc de montrer que le décalage $\mathbf{T}_{\tilde{S}}$ est sofique, ce à quoi nous allons nous attarder.

Proposition 3.2.7. *Le décalage $\Sigma = \mathbf{SA}_{(1,0,\dots,0)\mathbb{Z}}(\pi_V(\mathbf{T}_{\tilde{S}}))$ est effectif.*

Démonstration. Comme nous l'avons vu précédemment (voir la Proposition 2.2.1), la classe des décalage effectifs est stable par sous-action projective. Il nous suffit donc de montrer que $\mathbf{T}_{\tilde{\mathcal{S}}}$ est un décalage effectif. La suite de substitutions \mathbf{S} étant une suite effective, il en est de même pour la suite $\tilde{\mathbf{S}}$. Nous pouvons donc décrire un algorithme qui calcule les $\tilde{\mathbf{S}}$ -motifs, ce qui prouve que le décalage $\mathbf{T}_{\tilde{\mathcal{S}}}$ est effectif. \square

Par le Théorème 3.1.1, il existe un décalage de type fini de dimension d sur un alphabet B , que l'on notera \mathbf{T}_{Σ} , et une fonction de bloc $\pi_{\Sigma} : B^{\mathbb{Z}^d} \rightarrow \mathcal{S}^{\mathbb{Z}^d}$ tels que $\Sigma = \pi_{\Sigma}(\mathbf{SA}_{(1,0,\dots,0)\mathbb{Z}}(\mathbf{T}_{\Sigma}))$. Il est à noter que le fait que $d \geq 2$ est ici décisif, puisque l'affirmation précédente n'est plus vraie pour $d = 1$.

Si nous considérons une configuration du décalage $\mathbf{T}_{\tilde{\mathcal{S}}}$ défini dans la Partie 2.1.4, toute substitution qui y apparaît peut être choisie dans l'ensemble \mathcal{S} , à condition qu'elle soit compatible avec la configuration. Mais des substitutions différentes peuvent apparaître sur un même niveau. Ceci n'est pas cohérent avec la définition des décalages \mathbf{S} -adiques. Pour que cela le devienne, il suffit d'assurer que dans une configuration $x \in \mathbf{T}_{\tilde{\mathcal{S}}}$, la même substitution apparaît sur chaque ligne de $\pi_V(x)$ et la même substitution apparaît sur chaque colonne de $\pi_H(x)$. Nous définissons le décalage $\tilde{\mathbf{T}}_{\tilde{\mathcal{S}}}$ par :

$$\tilde{\mathbf{T}}_{\tilde{\mathcal{S}}} = \{x \in \mathbf{T}_{\tilde{\mathcal{S}}} : \forall (i, j) \in \mathbb{Z}^2, \pi_H(x)_{(i,j)} = \pi_H(x)_{(i,j+1)} \text{ et } \pi_V(x)_{(i,j)} = \pi_V(x)_{(i+1,j)}\},$$

et les conditions imposées permettent de déduire que :

$$\tilde{\mathbf{T}}_{\tilde{\mathcal{S}}} = \bigcup_{\tilde{\mathbf{S}} \in \tilde{\mathcal{S}}^{\mathbb{N}}} \mathbf{T}_{\tilde{\mathbf{S}}} \subset \mathbf{T}_{\tilde{\mathcal{S}}}.$$

Considérons finalement le décalage suivant :

$$\mathbf{T}_{\text{Final}} = \{(x, s) \in \tilde{\mathbf{T}}_{\tilde{\mathcal{S}}} \times \mathbf{T}_{\Sigma} : \forall (i, j) \in \mathbb{Z}^2, \pi_V(x)_{(i,j)} = \pi_{\Sigma}(s)_{(i,j)}\}.$$

Par le Corollaire 2.1.2, nous savons que le décalage $\mathbf{T}_{\tilde{\mathcal{S}}}$ est sofique, il est en donc de même pour $\tilde{\mathbf{T}}_{\tilde{\mathcal{S}}}$. Par construction, $\mathbf{T}_{\text{Final}}$ est aussi un décalage sofique. Considérons la fonction 1-bloc $\pi_{\text{Final}} : \mathbf{T}_{\text{Final}} \rightarrow A^{\mathbb{Z}^d}$ qui ne conserve que les lettres de l'alphabet A .

Proposition 3.2.8. $\pi_{\text{Final}}(\mathbf{T}_{\text{Final}}) = \mathbf{T}_{\tilde{\mathcal{S}}}$

Démonstration. Étant donnée une configuration $x \in \mathbf{T}_{\tilde{\mathcal{S}}}$, il est facile de construire l'élément correspondant dans $\mathbf{T}_{\text{Final}}$. Réciproquement, soit $x_{\text{Final}} \in \mathbf{T}_{\text{Final}}$. Quitte à remplacer les substitutions de l'ensemble \mathcal{S} par la composition de

deux substitutions de \mathcal{S} , nous pouvons supposer que pour toute substitution $s \in \mathcal{S}$ et pour toute lettre $a \in A$, $\mathbf{k}_1^s(a), \mathbf{k}_2^s(a) \geq 2$ (car les substitutions ne sont pas dégénérées).

La partie correspondant à $\tilde{\mathbf{T}}_{\mathcal{S}}$ dans la configuration x_{Final} nous assure que $\pi_{\text{Final}}(x_{\text{Final}})$ est un élément de l'un des décalages $\mathbf{T}_{\tilde{S}}$ pour une certaine suite $\tilde{S}' \in \tilde{S}^{\mathbb{N}}$. De plus, la condition liant les parties correspondant à $\tilde{\mathbf{T}}_{\mathcal{S}}$ avec celle correspondant à \mathbf{T}_{Σ} garantit que $S' = S$. Ainsi la substitution s_0 est la seule qui soit systématiquement répétée au moins deux fois, car nous avons supposé que $\mathbf{k}_1^s, \mathbf{k}_2^s \geq 2$. Le même raisonnement appliqué à une pré-image de x_{Final} par s_0 nous permet de retrouver la substitution s_1 , et ainsi de suite pour les autres éléments de la suite S . \square

Nous avons donc construit un décalage sofique qui permet de retrouver, par application de l'opération facteur **Fact**, le décalage S -adique \mathbf{T}_S à condition que la suite S soit effective, ce qui prouve le théorème. \square

Une réciproque ?

Dans cette partie nous cherchons une réciproque à l'énoncé du Théorème 3.2.5. Que peut-on dire d'un décalage effectif qui est S -adique ?

Dans la Partie 2.3.2, nous avons défini la notion de dérivation unique pour une substitution. Nous adaptons cette notion à un ensemble de substitutions. On dit que l'ensemble de substitutions \mathcal{S} est à *dérivation unique* si pour tout élément $x \in \tilde{\mathbf{T}}_{\mathcal{S}} = \bigcup_{S \in \mathcal{S}^{\mathbb{N}}} \mathbf{T}_S$, il existe une unique substitution $s \in \mathcal{S}$, une unique configuration $y \in A^{\mathbb{Z}^d}$ et un unique vecteur $i \in \bigcup_{a \in A} \mathbb{U}_{\mathbf{k}^s(a)}$ tels que $s_{\infty}(y) = \sigma^i(x)$.

Théorème 3.2.9. *Soit \mathcal{S} un ensemble de substitutions à dérivation unique, et soit $S \in \mathcal{S}^{\mathbb{N}}$ une suite de substitutions. Si le décalage S -adique \mathbf{T}_S est effectif, alors la suite S est effective.*

Remarque. L'énoncé du Théorème 3.2.9 est en particulier valable lorsque le décalage S -adique \mathbf{T}_S est sofique.

Démonstration. Comme le décalage \mathbf{T}_S est effectif, il existe une machine de Turing \mathcal{M} qui énumère tous ses motifs interdits (le complémentaire de son langage). Pour toute substitution s de l'ensemble \mathcal{S} , notons $\mathcal{E}_s = \{s(a) : a \in A\}$. Nous cherchons à déterminer dans un premier temps laquelle des substitutions $s \in \mathcal{S}$ est la première de la suite S . Pour cela, pour chacune de ces substitutions, nous essayons de partitionner l'espace \mathbb{Z}^d avec des motifs provenant tous

du même ensemble \mathcal{E}_s , de sorte qu'aucun motif énuméré par la machine \mathcal{M} n'apparaisse. Tous ces calculs peuvent être menés en parallèle par une même machine de Turing, et grâce à la condition de dérivation unique pour l'ensemble \mathcal{S} , nous sommes assurés que tous les calculs sauf un vont prendre fin au bout d'un temps fini (un calcul prend fin lorsque la machine est assurée qu'une telle partition n'existe pas). La machine s'arrête lorsque toutes les substitutions sauf une ont été rejetées, ce qui arrive toujours au bout d'un temps fini, et la substitution restante est s_0 . Nous pouvons alors ré-appliquer ce procédé à une pré-image de x par s_0 pour trouver s_1 , et ainsi de suite. La suite de substitution S est donc effective, ce qui prouve le théorème. \square

4 Décalages d'arbres

4.1 Généralités	94
4.1.1 Automate d'arbres	94
4.1.2 Décalages d'arbres irréductibles	99
4.1.3 Contexte, automate réduit et bloc synchronisant	101
4.1.4 Automates minimaux	103
4.1.5 Théorème de décomposition	108
Fonctions d'éclatement et de fusion	108
Énoncé et preuve du théorème	109
4.2 Décalages de type fini	111
4.2.1 Décalages de sommets et de transitions	111
4.2.2 Conjugaison des décalages de type fini	112
4.3 Les décalages presque de type fini (AFT)	116
4.3.1 Définition	116
4.3.2 Caractérisation des AFT par leur couverture de Shannon	119
4.4 Procédures de décision	120
4.4.1 Calcul de la couverture de Shannon	120
4.4.2 Automate des paires et graphe des paires	121
4.4.3 Localité	125
4.4.4 Automate fermant à gauche	126

Ce quatrième chapitre est consacré aux décalages d'arbres, et plus particulièrement à la classe des décalages d'arbres sofiques. La Partie 4.1 présente les outils pour étudier les décalages d'arbres sofiques : automate d'arbres et automate minimal. La Partie 4.1.5 est consacrée au théorème de décomposition, grâce auquel nous montrons que la conjugaison des décalages d'arbres de type fini est un problème décidable. La Partie 4.3 étudie la classe des décalages d'arbres presque de type fini, classe intermédiaire entre la classe des décalages d'arbres de type fini et la classe des décalages d'arbres sofiques.

4.1 Généralités

Dans cette partie nous nous intéressons aux décalages définis sur le monoïde libre à d générateurs \mathbb{M}_d . Comme \mathbb{M}_d est un monoïde libre son graphe de Cayley ne contient aucun cycle, c'est pourquoi on parle de décalage d'arbres dans cette partie. Rappelons que nous nous donnons un alphabet fini A .

Commençons par fixer les notations et le vocabulaire spécifiques aux décalages d'arbre. On considère l'alphabet $\{0, 1, \dots, d-1\}$, de sorte que chaque élément du monoïde \mathbb{M}_d peut être identifié à un mot sur $\{0, 1, \dots, d-1\}$. Un élément de $\{0, 1, \dots, d-1\}^*$, l'ensemble des mots sur l'alphabet $\{0, 1, \dots, d-1\}$, sera un *nœud*, et le nœud correspondant à l'élément neutre du monoïde, ou de manière équivalente au mot vide, sera appelé la *racine* et noté par ε . Si x est un nœud et que i est un élément de $\{0, 1, \dots, d-1\}$ alors on dit que $y = xi$ est le $i^{\text{ième}}$ fils de x , et que x est le père de y . Un *arbre* est une configuration, c'est-à-dire un élément de $A^{\{0,1,\dots,d-1\}^*}$; les arbres seront le plus souvent notés par t . Remarquons que les arbres considérés ici sont de rang fixe : tous les nœuds ont le même nombre de fils. Si t est un arbre et x un mot de $\{0, 1, \dots, d-1\}^*$, on rappelle que t_x désigne la lettre qui apparaît en position x dans l'arbre t . Le support élémentaire de taille n est formé de l'ensemble des mots sur $\{0, 1, \dots, d-1\}$ de longueur au plus n . Un bloc désigne un motif élémentaire, et si u est un bloc on note $|u|$ sa hauteur. On rappelle que si \mathbf{T} est un décalage d'arbres, son langage d'ordre n , noté $\mathcal{L}_n(\mathbf{T})$, est formé de tous les blocs élémentaires qui apparaissent dans des configurations de \mathbf{T} . Les nœuds d'un bloc de taille n correspondant à un mot de longueur n sur $\{0, 1, \dots, d-1\}$ sont appelés les *feuilles*, ce sont des nœuds sans fils.

Sans perte de généralité nous allons supposer dans toute la suite que les arbres considérés sont des arbres binaires, c'est-à-dire que nous nous restreignons au cas où $d = 2$. Dans ce cas nous parlerons de fils gauche et droit. On utilisera la notation (a, t_0, t_1) pour désigner l'arbre étiqueté par la lettre a à la racine, dont le fils gauche est t_0 et le fils droit t_1 . En particulier, un bloc de hauteur 1 sera noté (a, b, c) , où a étiquette la racine, b le fils gauche de la racine et c le fils droit de la racine.

4.1.1 Automate d'arbres

Dans cette partie nous présentons une version des automates finis d'arbres qui reconnaît des configurations infinies ou des blocs finis. Dans ce modèle le calcul remonte des branches infinies vers la racine. L'acceptation ou non d'un arbre est déterminée simplement par l'existence d'un calcul de l'automate remontant jusqu'à la racine, car tous les états sont choisis terminaux. Cette version d'auto-

mate d'arbres correspond à la version standard d'automates d'arbres montants, lorsque tous les états sont à la fois initiaux et terminaux. Pour d'autres version d'automates d'arbres le lecteur pourra se référer à [CDG⁺07].

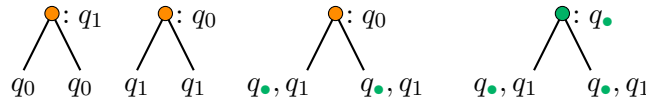
Définition 15. Un *automate fini d'arbres* est une structure $\mathcal{A} = (V, A, \Delta)$ où V est un ensemble fini d'états (ou sommets), A est un alphabet fini, et Δ est un ensemble de transitions de la forme $(q_0, q_1), a \rightarrow q$, avec $q, q_i \in V, a \in A$. Une transition $(q_0, q_1), a \rightarrow q$ est dite étiquetée par a , sortant du couple d'états (q_0, q_1) et arrivant dans l'état q .

Remarque. Dans toute la suite on omettra l'adjectif fini, et on parlera simplement d'automate d'arbres pour désigner un automate fini d'arbres.

Exemple 4.1.1. Soit l'alphabet à deux éléments $A = \{ \bullet, \circ \}$. On définit un automate \mathcal{A}_1 dont l'ensemble d'états est $Q_1 = \{q_0, q_1\}$, donné par l'ensemble de transitions suivantes :

$$\Delta_1 = \{(q_0, q_0, \bullet) \rightarrow q_1, (q_1, q_1, \circ) \rightarrow q_0\}.$$

Exemple 4.1.2. Avec l'alphabet $A = \{ \circ, \bullet \}$, toujours sur des arbres binaires, on définit un automate \mathcal{A}_2 dont l'ensemble d'états est $Q_2 = \{q_0, q_1, q_\bullet\}$, donné par l'ensemble de transitions suivantes :

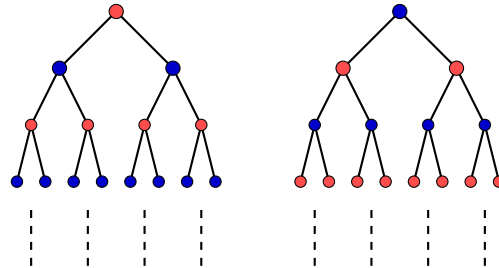


Un tel automate est dit *déterministe* si pour tout couple d'états (q_0, q_1) et pour toute lettre $a \in A$, il existe au plus une transition $(q_0, q_1), a \rightarrow q$. L'ensemble des transitions définit alors une fonction partielle $\delta : V^2 \times A \rightarrow V$. Dans ce cas, on utilisera aussi pour l'automate la notation $\mathcal{A} = (V, A, \delta)$.

Un *calcul* de l'automate $\mathcal{A} = (V, A, \Delta)$ sur un arbre t est un arbre C sur l'alphabet V tel que pour tout nœud x , s'il existe une transition $(C_{x0}, C_{x1}, t_x) \rightarrow C_x \in \Delta$. Un arbre t est *accepté* par un automate \mathcal{A} s'il existe un calcul de \mathcal{A} sur t . Il est aisé de voir que l'ensemble des arbres acceptés par un automate \mathcal{A} constitue un décalage, que l'on note $\mathbf{T}_{\mathcal{A}}$.

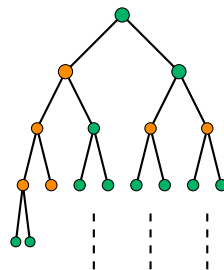
Un arbre fini est dit *complet* si chacun de ses nœuds a soit deux fils, soit aucun fils. On définit une notion de calcul d'un automate d'arbres sur un arbre fini complet de la manière suivante. Un *calcul fini* de l'automate $\mathcal{A} = (V, A, \Delta)$ sur un arbre fini complet u est un arbre fini complet C sur l'alphabet V tel que, pour tout nœud x de u , il existe une transition $(C_{x0}, C_{x1}), u_x \rightarrow C_x \in \Delta$.

Exemple 4.1.3. Les seuls arbres acceptés par l'automate de l'Exemple 4.1.1 sont les deux arbres suivants :



Sur chacun des arbres, tous les nœuds d'un même niveau partagent la même couleur et deux niveaux consécutifs ont des couleurs différentes. Le décalage d'arbres $\mathbf{T}_{\mathcal{A}_1}$ est de type fini.

Exemple 4.1.4. Les arbres acceptés par l'automate de l'Exemple 4.1.2 sont ceux qui ne contiennent aucun chemin le long duquel on trouve un nombre pair de \bullet entre deux \bullet . Le décalage d'arbres $\mathbf{T}_{\mathcal{A}_2}$ est sofique mais pas de type fini.



Soit m un entier positif. Un automate d'arbres déterministe \mathcal{A} est m -local si pour toutes paires d'arbres (t, t') contenant un même bloc b de hauteur m en position x dans t et en position x' dans t' , et pour tous calculs C sur t et C' sur t' de l'automate \mathcal{A} , on a $C_x = C'_{x'}$.

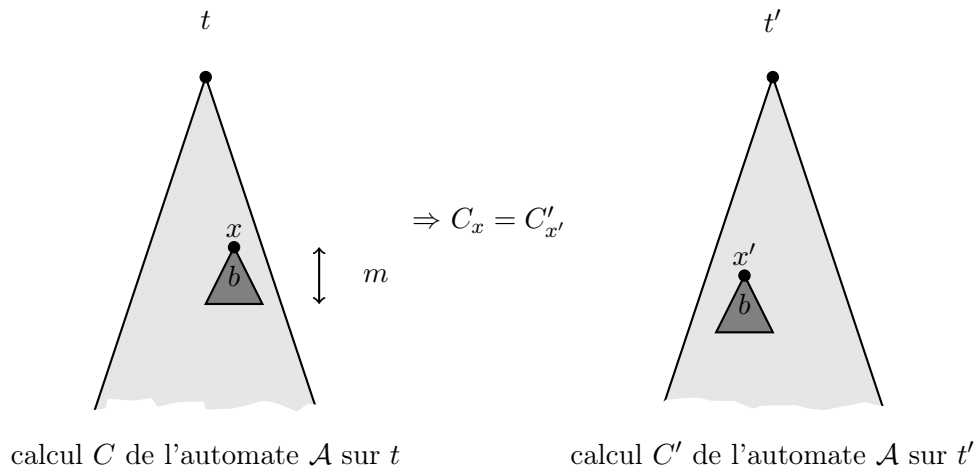


FIGURE 27: Notion de m -localité pour un automate d'arbres \mathcal{A} , illustrée sur deux calculs de \mathcal{A} sur les arbres t et t' dans lesquels un même bloc b de hauteur m apparaît.

Autrement dit une mémoire de hauteur m suffit à déterminer l'état atteint par n'importe quel calcul C de l'automate. On dit dans ce cas que le bloc b *force* l'état C_x dans l'automate \mathcal{A} . On dit simplement qu'un automate d'arbres est *local* s'il est m -local pour un entier positif m .

Si un automate d'arbres $\mathcal{A} = (Q, A, \delta)$ est déterministe, on peut définir par induction une fonction δ^k sur les blocs de hauteur k

- si $k = 0$ alors $\delta^0 = \delta$;
- si $k \geq 1$ alors $\delta^k((t_0, t_1, a)) = \delta(\delta^{k-1}(t_0), \delta^{k-1}(t_1), a)$.

Ces fonctions δ^k seront aussi appelées δ quand aucune confusion n'est possible sur la valeur de l'entier k à choisir.

Les automates d'arbres acceptent exactement les décalages d'arbres sofique, et les automates d'arbres locaux accepte exactement les décalages d'arbres de type fini. Les démonstrations de ces caractérisations sont similaires à celles des résultats correspondants pour les décalages sur \mathbb{N} ou sur \mathbb{Z} (voir [LM95] ou [Kit98]).

Proposition 4.1.1 ([AB09]). *Tout décalage d'arbres de type fini est accepté par un automate d'arbre déterministe local. Réciproquement tout décalage d'arbres accepté par un automate d'arbres déterministe local est de type fini.*

Démonstration. Soit \mathbf{T}_F un décalage d'arbres de type fini défini par l'ensemble fini de motifs interdits F . Sans perte de généralité, on peut supposer que les motifs de l'ensemble F sont tous des blocs de hauteurs m avec m entier strictement positif. On définit un automate d'arbres déterministe $\mathcal{A} = (V, A, \delta)$ dont les états sont les blocs du langage d'ordre m : $V = \mathcal{L}_m(X)$. Pour $p_i \in V, a \in A$, si le bloc $q = (a, p_0, p_1)$ de hauteur $m + 1$ est un bloc autorisé de \mathbf{T} , alors $\delta(p_0, p_1, a) = \text{tronc}(q)$, où $\text{tronc}(q)$ est le bloc de hauteur m tel que $\text{tronc}(q)_x = q_x$ pour $x \in \{0, 1\}^{\leq m-1}$. La fonction partielle δ n'est pas définie sinon. Par construction, l'automate \mathcal{A} est déterministe et m -local. Il est clair que cet automate reconnaît le décalage \mathbf{T} , ce qui prouve la première partie de la proposition.

Soit \mathbf{T} un décalage d'arbres et $\mathcal{A} = (V, A, \delta)$ un automate d'arbre m -local qui accepte \mathbf{T} . On définit F comme l'ensemble des blocs interdits de hauteur $m + 1$ dans \mathbf{T} . Il est aisé de voir que $\mathbf{T} \subseteq \mathbf{T}_F$. Supposons maintenant que $t \in \mathbf{T}_F$. On définit un calcul de l'automate \mathcal{A} sur le bloc t de la manière suivante. Pour tout $x \in \{0, 1\}^*$, on appelle C_x l'état sur lequel termine tout calcul de l'automate \mathcal{A} sur le bloc de hauteur m enraciné en x dans l'arbre t . Soit b le bloc de hauteur $m + 1$ enraciné en x dans t . Comme $t \in \mathbf{T}_F$, b est un bloc autorisé pour \mathbf{T} , donc ce motif apparaît dans un arbre $t' \in \mathbf{T}$ en position $y \in \{0, 1\}^*$. Soit C' le calcul de l'automate \mathcal{A} sur t' . Ainsi $\delta(c_{y0}, \dots, c_{yd}, t'_y) = c_y$. Comme l'automate \mathcal{A} est m -local, $C_{xi} = C_{yi}$ pour $0 \leq i \leq d$ et $C_y = C_x$. Comme $t_x = t'_y$ il s'ensuit que $\delta(C_{x0}, \dots, C_{xd}, t_x) = C_x$. Ainsi c est un calcul de l'automate \mathcal{A} sur t et $t \in \mathbf{T}$, ce qui achève la démonstration de la proposition. \square

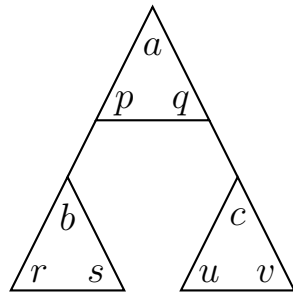
Proposition 4.1.2. *Un décalage d'arbres est sofique si et seulement s'il est reconnu par un automate d'arbres déterministe.*

Démonstration. Soit \mathbf{T} un décalage sofique sur l'alphabet A . Alors \mathbf{T} est l'image d'un décalage de type fini \mathbf{T}' par une m -fonction de bloc Φ . Soit $V = \mathcal{L}_m(\mathbf{T}')$ l'ensemble des blocs de hauteur m qui apparaissent dans \mathbf{T}' . On définit l'automate d'arbres $\mathcal{A} = (V, A, \delta)$, dont les transitions sont données par la règle suivante : $(p, q, a) \rightarrow r \Leftrightarrow a = \phi(r)$ où ϕ est la fonction locale qui définit Φ . Alors par définition cet automate est déterministe et reconnaît bien le décalage \mathbf{T} .

Réciproquement soit $\mathcal{A} = (V, A, \delta)$ un automate d'arbres déterministe qui accepte un décalage d'arbres $\mathbf{T}_\mathcal{A}$. On définit un nouvel alphabet

$$A' = \{(p, q, a) \in V^2 \times A \mid \exists r \in V \text{ tel que } \delta(p, q, a) = r \text{ dans } \mathcal{A}\},$$

et sur cet alphabet A' on définit le décalage d'arbres de type fini \mathbf{T}' dont les motifs autorisés sont de la forme :



avec $p = \delta(r, s, b)$ et $q = \delta(u, v, c)$

On appelle Φ la 1-fonction de bloc qui à un arbre t' sur A' associe l'arbre t sur A obtenu en ne conservant que la partie dans A de chaque lettre de A' . Alors le décalage $\Phi(\mathbf{T}')$ est un décalage d'arbre sofique comme image par une fonction de bloc d'un décalage d'arbres de type fini, et $\Phi(\mathbf{T}')$ est exactement \mathbf{T} ce qui achève la démonstration. \square

4.1.2 Décalages d'arbres irréductibles

Un *code préfixe fini et complet* de $\{0, 1\}^*$ est un ensemble préfixe (i.e. aucun mot n'est préfixe d'un autre mot) P de mots finis sur $\{0, 1\}^*$ tel que tout mot de $\{0, 1\}^*$ de longueur supérieure à celles des mots de P possède un préfixe dans P . Un décalage d'arbres \mathbf{T} est *irréductible* si pour toute paire de blocs $u, v \in \mathcal{L}(\mathbf{T})$, il existe un arbre $t \in \mathbf{T}$ et un code préfixe fini et complet $P \subset \{0, 1\}^{\geq |u|}$, tel que u est le sous-arbre de t à la racine ε , et v est un sous-arbre de t en position x pour tout $x \in P$.

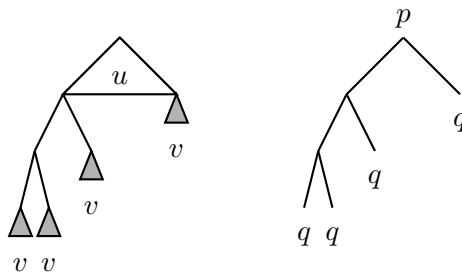


FIGURE 28: Décalage d'arbres irréductible et automate d'arbres irréductible.

Un automate d'arbres est *irréductible* si pour toute paire d'états p, q il existe un code préfixe fini et complet $P \subset \{0, 1\}^*$ et un calcul fini C de l'automate sur un motif u tel que $C_\varepsilon = p$ et $C_x = q$ pour tout $x \in P$. On dit dans ce cas qu'il existe un *hyper-chemin de q à p* étiqueté par u . Pour deux états p, q d'un automate, on dit que p est *accessible* depuis q s'il existe un hyper-chemin de q à p .

Proposition 4.1.3. *Un automate d'arbres irréductible accepte un décalage d'arbres sofique irréductible.*

Démonstration. Soit \mathcal{A} un automate d'arbres irréductible. On appelle $\mathbf{T}_{\mathcal{A}}$ le décalage d'arbres sofique accepté par l'automate \mathcal{A} . Soient u et v deux blocs dans le langage de $\mathbf{T}_{\mathcal{A}}$, et on suppose que le bloc u est de hauteur k . Alors il existe un arbre $t_u \in \mathbf{T}_{\mathcal{A}}$ dans lequel u apparaît à la racine, et un arbre $t_v \in \mathbf{T}_{\mathcal{A}}$ dans lequel v apparaît à la racine. Comme $t_u \in \mathbf{T}_{\mathcal{A}}$ il existe un calcul C de \mathcal{A} sur t_u , dans lequel les feuilles du motifs p sont étiquetées par des états p_i pour $0 \leq i \leq 2^{k-1}$. De même il existe un calcul C' de \mathcal{A} sur t_v dans lequel $C'_\varepsilon = q$. Comme \mathcal{A} est irréductible, il existe un hyper-chemin de p_i à q étiqueté par w_i , et associé à un code préfixe fini et complet $P_i \subset \{0, 1\}^{\geq |w_i|}$. On définit le code préfixe fini complet $P = \bigcup_{0 \leq i \leq 2^{k-1}} w_i P_i$. On définit l'arbre t de la manière suivante :

- le bloc u apparaît à la racine de t ;
- le bloc w_i apparaît à la $i^{\text{ème}}$ feuille de u ;
- l'arbre t_v est enraciné en chaque feuille des w_i .

Cet arbre est accepté par l'automate \mathcal{A} donc appartient bien au décalage \mathbf{T} . Ceci montre que \mathbf{T} est irréductible. \square

La réciproque de la Proposition 4.1.3 peut être démontrée en utilisant la notion de couverture de Shannon d'un décalage sofique, définie dans la Partie 4.1.4. Ainsi les notions d'irréductibilité pour les décalages et pour les automates sont directement corrélés.

4.1.3 Contexte, automate réduit et bloc synchronisant

Les automates d'arbres reconnaissent les décalages d'arbres soifiques, mais pour l'instant nous n'avons pas trouvé d'automate d'arbres canonique reconnaissant un décalage d'arbres \mathbf{T} . Dans ce but, nous définissons la notion de contexte d'un bloc fini, qui correspond à la notion de *follower set* de Lind et Marcus [LM95] pour les décalages sur \mathbb{N} ou \mathbb{Z} .

Soit \mathbf{T} un décalage d'arbre. Un *contexte* c est un motif fini dont une des feuilles joue un rôle spécial, on dira qu'elle est *marquée*. Si u est un motif, $c(u)$ est le motif obtenu en remplaçant la feuille marquée du contexte c par le motif u . Si $c(u)$ apparaît dans un motif du langage $\mathcal{L}(\mathbf{T})$, on dit alors que c est un *contexte du motif u dans \mathbf{T}* . Étant donné un bloc u , on note par $\text{cont}_{\mathbf{T}}(u)$ l'ensemble des tous les contextes de u dans \mathbf{T} . Étant donné un automate d'arbres $\mathcal{A} = (V, A, \Delta)$ qui accepte un décalage d'arbres sofique \mathbf{T} , le *contexte* d'un état $q \in V$ dans \mathcal{A} est l'ensemble des motifs u avec une feuille marquée en position x sur lesquels il existe un calcul fini C de l'automate \mathcal{A} avec $C_x = q$. On le note $\text{cont}_{\mathcal{A}}(q)$. Remarquons que le contexte d'un motif u dans \mathbf{T} est l'union des contextes des états p qui terminent un calcul de \mathcal{A} sur u . Un décalage d'arbres sofique a donc un nombre fini de contextes distincts.

Soit $\mathcal{A} = (V, A, \Delta)$ un automate d'arbre. On dit que l'automate \mathcal{A} est *réduit* si $p \neq q$ implique $\text{cont}_{\mathcal{A}}(p) \neq \text{cont}_{\mathcal{A}}(q)$ pour $p, q \in V$, autrement dit si deux états distincts ont des contextes distincts. Si $\mathcal{A} = (V, A, \Delta)$ et $\mathcal{A}' = (V', A, \Delta')$ sont deux automates d'arbres déterministes, une *réduction* de \mathcal{A} vers \mathcal{A}' est une fonction h de V vers V' telle que, pour toute lettre $a \in A$, on a $p, q, a \rightarrow r \in \Delta$ si et seulement si $(h(p), h(q), a) \rightarrow r \in \Delta'$. Si \mathcal{A}' est un automate obtenu par réduction à partir d'un automate \mathcal{A} , alors \mathcal{A} et \mathcal{A}' définissent le même décalage.

Soit \mathbf{T} un décalage d'arbres sofique déterministe accepté par un automate d'arbres $\mathcal{A} = (V, A, \Delta)$. On définit un automate d'arbres déterministe et réduit $R(\mathcal{A})$ qui accepte \mathbf{T} , et que l'on appelle *réduction* de l'automate \mathcal{A} . Les états de $R(\mathcal{A})$ sont les classes de la partition la plus grossière de l'ensemble V telle que si les états p, q appartiennent à la même classe, alors pour toute lettre $a \in A$ et pour tout état $r \in Q$, $\delta(p, r, a)$ et $\delta(q, r, a)$ (resp. $\delta(r, p, a)$ et $\delta(r, q, a)$)

appartiennent à la même classe, et la transition $\delta(p, r, a)$ (resp. $\delta(r, p, a)$) est définie si et seulement si la transition $\delta(q, r, a)$ (resp. $\delta(r, q, a)$) est définie. On note par $[p]$ la classe de l'état p . La transition $([p], [q], a \rightarrow [\delta(p, q, a)])$ est une transition de l'automate $R(\mathcal{A})$ si et seulement si $\delta(p, q, a)$ est définie. Cette définition ne dépend pas du choix du représentant de la classe et est donc bien fondée. Par définition, l'automate $R(\mathcal{A})$ est réduit, et la fonction h définie par $h(p) = [p]$ est une réduction de \mathcal{A} vers $R(\mathcal{A})$.

L'algorithme de minimisation pour les automates d'arbres déterministes qui acceptent des blocs finis, que l'on peut trouver dans [CDG⁺07, Section 1.5], peut être appliqué pour calculer une réduction d'un automate d'arbres acceptant le même décalage d'arbre.

Soit $\mathcal{A} = (V, A, \Delta)$ un automate d'arbres déterministe qui reconnaît un décalage d'arbres sofique \mathbf{T} , et soit u un bloc (resp. un motif). On dit que u est un *bloc synchronisant* (resp. un *motif synchronisant*) de l'automate \mathcal{A} s'il existe au moins un calcul fini de l'automate \mathcal{A} sur u , et si de plus tous les calculs finis de \mathcal{A} sur u terminent dans un même état $q \in Q$. On dit alors que le bloc u *synchronise* sur l'état q . Un automate d'arbres déterministe qui possède un bloc synchronisant est appelé *automate synchronisé*.

Remarque. Si u est un bloc synchronisant, alors tout motif dans lequel u apparaît à la racine est synchronisant.

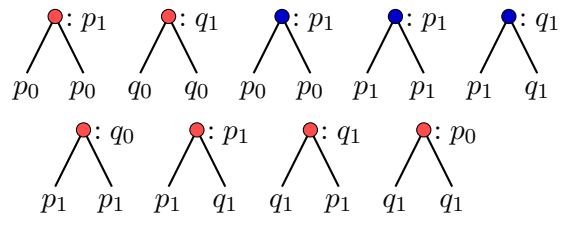
Soit $\mathcal{A} = (V, A, \Delta)$ un automate d'arbre. On définit *l'automate déterminisé de \mathcal{A}* , noté $\mathcal{D}(\mathcal{A})$, comme la partie accessible depuis l'état V dans l'automate d'arbres $(\mathfrak{P}(V), A, \delta')$, avec pour $P, Q \in \mathfrak{P}(V)$, $\delta'(P, Q, a) = \{\delta(p, q, a) \mid p \in P, q \in Q\}$ si l'ensemble est non vide, et n'est pas défini sinon.

Proposition 4.1.4. *Un automate \mathcal{A} est synchronisé si et seulement si son automate déterminisé $\mathcal{D}(\mathcal{A})$ possède un état qui est un singleton.*

La démonstration de la Proposition 4.1.4 découle directement de la définition d'un bloc synchronisant.

Proposition 4.1.5. *Il existe des automates d'arbres déterministes, irréductibles et réduits qui ne sont pas synchronisés.*

Démonstration. Soit \mathbf{T} le décalage d'arbres plein sur l'alphabet à deux éléments $A = \{\bullet, \circ\}$. Il est trivialement accepté par un automate d'arbres à un seul état. Mais il est aussi accepté par l'automate d'arbres déterministe et réduit $\mathcal{A} = (V, A, \delta)$ dont les règles de transitions sont présentées ci-dessous.

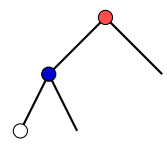


L'automate A est irréductible : il existe un hyper-chemin

$$p_0 \rightsquigarrow p_1 \rightsquigarrow q_0 \rightsquigarrow q_1 \rightsquigarrow p_0,$$

où $p \rightsquigarrow q$ désigne un hyper-chemin de l'état p vers l'état q .

L'automate A est réduit, car deux états différents ont des contextes différents. En effet, les états q_0, q_1 n'ont pas même contexte que les états p_0, p_1 , car le motif dont la racine est étiquetée par \bullet , et dont la feuille gauche est marquée est un contexte des états p_0, p_1 mais pas de q_0, q_1 . De plus, le motif suivant, dans lequel la feuille marquée est représentée par un \circ :



est un contexte de p_0, q_1 mais pas de p_1, q_0 .

L'automate $\mathcal{D}(A)$ contient deux états $V = \{p_0, q_0, p_1, q_1\}$ et $\{p_1, q_1\}$, mais aucun état réduit à un singleton et donc \mathcal{A} n'est pas synchronisé et la Proposition 4.1.5 est démontrée. \square

En particulier, la Proposition 4.1.5 nous apprend que, contrairement au cas des décalages sur \mathbb{N} ou \mathbb{Z} , un décalage d'arbres sofique peut être accepté par plusieurs automates d'arbre irréductibles, déterministes et réduits.

4.1.4 Automates minimaux

Comme nous venons de le voir dans la Partie 4.1.3, les décalages d'arbres différent des décalages unidimensionnels au moins par le fait qu'il existe des automates d'arbres réduits, irréductibles et déterministes qui ne possèdent pas de bloc synchronisant (voir la Proposition 4.1.5). Il est cependant possible de construire un tel automate en ajoutant la condition qu'il soit synchronisé. Dans

cette partie nous présentons deux automates minimaux, la couverture de Shannon (ou couverture de Fischer) et la couverture de Krieger, qui reposent tous les deux sur la notion de contexte d'un arbre (contexte d'un arbre fini pour la couverture de Shannon, et contexte d'un arbre infini pour la couverture de Krieger).

Soit \mathbf{T} un décalage d'arbres sofique. L'automate des contextes du décalage \mathbf{T} est l'automate d'arbres déterministe $\mathcal{C} = (V, A, \Delta)$, où V est l'ensemble des contextes non vides des blocs finis apparaissant dans le décalage \mathbf{T} . Dans le cas d'un décalage d'arbres \mathbf{T} sofique, l'ensemble V est fini, mais ce n'est pas le cas lorsque \mathbf{T} n'est pas sofique. Les transitions de l'automate \mathcal{C} sont de la forme $(\text{cont}_{\mathbf{T}}(u), \text{cont}_{\mathbf{T}}(v)), a \rightarrow \text{cont}_{\mathbf{T}}(a, u, v)$, avec $u, v \in \mathcal{L}(\mathbf{T})$.

Remarque. Il est possible de définir un automate des contextes pour un décalage d'arbre qui n'est pas sofique, mais dans ce cas cet automate possède une infinité d'états.

Proposition 4.1.6. *L'automate des contextes d'un décalage d'arbres sofique est synchronisé.*

Démonstration. Soit \mathcal{C} l'automate des contextes d'un décalage d'arbres sofique \mathbf{T} . Il existe un calcul fini C_1 de l'automate \mathcal{C} sur un certain bloc u_1 qui se termine dans l'état $\text{cont}_{\mathbf{T}}(u_1)$. Supposons que ce bloc u_1 n'est pas un bloc synchronisant de l'automate \mathcal{C} . Il existe donc un autre calcul C_2 de l'automate \mathcal{C} sur le bloc u_1 qui termine sur un état $\text{cont}_{\mathbf{T}}(u_2)$ pour un certain bloc u_1 tel que $\text{cont}_{\mathbf{T}}(u_2) \neq \text{cont}_{\mathbf{T}}(u_1)$. Ainsi $\text{cont}_{\mathbf{T}}(u_2) \subsetneq \text{cont}_{\mathbf{T}}(u_1)$, car $\text{cont}_{\mathbf{T}}(u_2) \neq \text{cont}_{\mathbf{T}}(u_1)$. Si le bloc u_2 n'est pas un bloc synchronisant de l'automate \mathcal{C} , on peut à nouveau appliquer le même raisonnement pour construire un bloc u_3 . En itérant ce procédé, nous obtenons soit un bloc synchronisant pour l'automate \mathcal{C} , soit une suite strictement décroissante de contextes. Et comme il n'existe qu'un nombre fini de contextes, cette deuxième option est à exclure, ce qui prouve l'existence d'un bloc synchronisant dans l'automate des contextes \mathcal{C} . \square

Si \mathcal{A} est un automate d'arbres, on dit qu'une composante de l'automate est *minimale* si tout hyper-chemin partant d'un état de la composante arrive aussi dans la composante (voir le chapitre [BBrEP10] pour la définition en dimension 1).

Proposition 4.1.7. *Si \mathbf{T} est un décalage d'arbres sofique irréductible, alors son automate des contextes \mathcal{C} possède une unique composante minimale irréductible \mathcal{S} , obtenue en ne conservant que les états accessibles depuis l'état $\text{cont}_{\mathbf{T}}(z)$, où z est un bloc synchronisant de \mathcal{C} . L'automate \mathcal{S} est la couverture de Shannon (aussi appelée couverture de Fischer) du décalage \mathbf{T} .*

Démonstration. Comme les états de l'automate \mathcal{S} sont les états accessibles depuis l'état $q = \text{cont}_{\mathbf{T}}(z)$, \mathcal{S} est une composante irréductible minimale. En effet, soit $p = \text{cont}_{\mathbf{T}}(u)$ un état accessible à partir de l'état q . Il existe par définition un hyper-chemin de q à p étiqueté par un motif v de \mathbf{T} , et le bloc u est synchronisant. Comme \mathbf{T} est un décalage irréductible, il existe un motif v et un code préfixe fini complet P formé de mots de hauteur au moins la hauteur de z , tel que z est enraciné à la racine de v et tel que u est un sous-arbre de v enraciné en tout nœud $x \in P$. Comme tout calcul de l'automate des contextes \mathcal{C} sur v se termine dans l'état p , il existe un hyper-chemin de p à q .

Montrons à présent que \mathcal{S} est unique. Si un état $p = \text{cont}_{\mathbf{T}}(u)$ appartient à une autre composante R irréductible et minimale de l'automate \mathcal{C} , par irréductibilité du décalage \mathbf{T} , il existe un motif w de \mathbf{T} et un code préfixe fini complet P formé de mots de hauteur au moins la hauteur de z , tel que z est enraciné à la racine de w et tel que u est un sous-arbre de w enraciné en tout nœud $x \in P$. Par définition de l'automate \mathcal{C} , il existe alors un hyper-chemin de p à q . Donc q appartient à R , ce qui implique que $R = \mathcal{S}$. Donc \mathcal{S} est l'unique composante minimale irréductible de \mathcal{C} . \square

On définit la *couverture de Shannon* (aussi appelée couverture de Fischer) d'un décalage d'arbres sofique irréductible \mathbf{T} comme l'unique composante maximale irréductible \mathcal{S} de son automate des contextes \mathcal{C} , obtenue en ne conservant que les états accessibles depuis l'état $\text{cont}_{\mathbf{T}}(z)$, où z est un bloc synchronisant de \mathcal{C} . Comme le bloc z est synchronisant et que l'automate \mathcal{C} est déterministe, tout état de cette composante est le contexte d'un bloc synchronisant. La définition de la couverture de Shannon ne dépend donc pas du choix du bloc synchronisant z .

Montrons que les états accessibles depuis $\text{cont}_{\mathbf{T}}(z)$ forment une composante irréductible, et que celle-ci est l'unique composante irréductible de l'automate \mathcal{C} . Il suffit pour cela de montrer qu'il existe un hyper-chemin depuis tout état de l'automate \mathcal{C} vers $\text{cont}_{\mathbf{T}}(z)$. Soit $p = \text{cont}_{\mathbf{T}}(u)$ un état de \mathcal{C} . Comme \mathbf{T} est irréductible, il existe un motif w de \mathbf{T} et un code préfixe fini complet P de $\{0, 1\}^{\geq \text{hauteur}(z)}$, tel que z est un sous-arbre de w enraciné en ε , et u est un sous-arbre de w enraciné en tout $x \in P$. Soit C un calcul de \mathcal{C} sur le motif w . Alors $C_\varepsilon = \text{cont}_{\mathbf{T}}(z)$, et pour toute feuille $x \in P$, $C_x = p$. Donc il existe un hyper-chemin de p vers $\text{cont}_{\mathbf{T}}(z)$. Enfin, il est facile de vérifier que l'automate \mathcal{S} accepte le décalage \mathbf{T} (cela découle directement de l'irréductibilité du décalage \mathbf{T}).

La couverture de Shannon d'un décalage sofique irréductible \mathbf{T} étant bien définie, nous montrons qu'il s'agit de l'unique automate d'arbres déterministe, irréductible et synchronisé qui accepte ce décalage \mathbf{T} .

Proposition 4.1.8. *Deux automates d'arbres réduits, déterministes, irréductibles et synchronisés acceptant le même décalage sofique irréductible sont égaux à un renommage des états près.*

Démonstration. Soient $\mathcal{A} = (V, A, \Delta)$ et $\mathcal{A}' = (V', A, \Delta')$ deux automates d'arbres réduits, déterministes, irréductibles et synchronisés acceptant le même décalage sofique irréductible \mathbf{T} . Soit $u \in \mathcal{L}_m(X)$ (resp. $v \in \mathcal{L}_m(X)$) un bloc synchronisant de l'automate \mathcal{A} (resp. \mathcal{A}'). Nous supposons que le bloc z synchronise sur l'état p dans l'automate \mathcal{A} . Comme \mathbf{T} est irréductible, il existe un motif z du décalage \mathbf{T} et un code préfixe fini complet $P \subset \{0, 1\}^{\geq m-1}$, tels que le bloc u est un sous-arbre du motif z enraciné en ε , et tels que v est un sous-arbre du motif z enraciné en toute feuille $x \in P$. Comme les automates \mathcal{A} et \mathcal{A}' sont déterministes, le motif z est un motif synchronisant pour les deux automates \mathcal{A} et \mathcal{A}' . Chaque calcul de l'automate \mathcal{A} sur le motif z termine sur l'état p , tandis que chaque calcul de l'automate \mathcal{A}' sur le motif z termine sur un état p' dans V' .

Nous définissons une bijection $\varphi : V \rightarrow V'$ de la manière suivante. Soit q un état de l'automate \mathcal{A} . Comme \mathcal{A} est irréductible, il existe un calcul C de \mathcal{A} sur un motif w , et un code préfixe P tels que $C_\varepsilon = q$ et $C_x = p$ pour tout $x \in P$. Puisque l'automate \mathcal{A} est déterministe et que le bloc z est un bloc synchronisant pour l'automate \mathcal{A} , tout calcul fini de l'automate \mathcal{A} sur w termine dans l'état q . Donc le contexte de q dans \mathcal{A} est le même que le contexte de w dans \mathbf{T} . De plus, tout calcul fini de l'automate \mathcal{A}' sur w termine dans un état q' tel que le contexte de q' dans \mathcal{A}' est le même que le contexte de w dans \mathbf{T} . Comme l'automate \mathcal{A} (resp. l'automate \mathcal{A}') est réduit, deux états avec le même contexte dans \mathcal{A} (resp. dans \mathcal{A}') sont en fait égaux. Donc si $q = \text{cont}_{\mathbf{T}}(w)$, $r = \text{cont}_{\mathbf{T}}(v)$, et si $(q, r), a \rightarrow s$ est une transition de Δ , nous avons $s = \text{cont}_{\mathbf{T}}(a, w, v)$. Dans ce cas nous définissons $\varphi(q) = q'$. Ainsi φ définit un isomorphisme entre \mathcal{A} et \mathcal{A}' , où φ est une bijection telle que $(q, r), a \rightarrow s$ est une transition de Δ si et seulement si $(\varphi(q), \varphi(r)), a \rightarrow \varphi(s)$ est une transition de Δ' . \square

Soit t un arbre infini sur l'alphabet A et c un contexte. On désigne par $c(t)$ le motif c dans lequel la feuille marquée du contexte c est remplacée par l'arbre t . Si le motif infini $c(t)$ apparaît dans un arbre du décalage \mathbf{T} , on dit que c est un *contexte de t dans \mathbf{T}* . Étant donné un arbre t sur l'alphabet A , on note par $\text{cont}_{\mathbf{T}}(t)$ l'ensemble des contextes de t dans \mathbf{T} . La *couverture de Krieger* d'un décalage d'arbres \mathbf{T} est l'automate d'arbre dont les états sont les ensembles non vides de la forme $\text{cont}_{\mathbf{T}}(t)$, et dont les transitions sont $(\text{cont}_{\mathbf{T}}(t), \text{cont}_{\mathbf{T}}(t'), a) \rightarrow \text{cont}_{\mathbf{T}}(a, t, t')$, où t, t' sont deux arbres sur l'alphabet A . Ainsi définie, la couverture de Krieger d'un décalage d'arbres \mathbf{T} est un automate d'arbres qui accepte \mathbf{T} . La couverture de Krieger diffère de l'automate des

contextes, car les contextes que l'on considère ici sont ceux d'arbres infinis, alors que les contextes de l'automate des contextes sont ceux d'arbres finis. Quel lien existe-t-il entre ces deux automates ? Et quel est le lien entre les couvertures de Shannon et de Krieger ? Nous allons montrer ces liens dans la Proposition 4.1.9, qui étend aux arbres une situation similaire pour les mots [BBrEP10].

On dit qu'un automate d'arbres $\mathcal{A} = (V, A, \Delta)$ est un *sous-automate* d'un automate d'arbres $\mathcal{A}' = (V', A, \Delta')$ sur le même alphabet A si $V \subseteq V'$ et si $\Delta = \Delta' \cap (V \times V \times A \times V)$. Un sous-automate \mathcal{A} est *minimal* si toute transition du sur-automate \mathcal{A}' partant d'un état de \mathcal{A} termine dans un état de \mathcal{A} .

Proposition 4.1.9. *La couverture de Krieger d'un décalage d'arbres sofique \mathbf{T} est, à isomorphisme près, un sous-automate de l'automate des contextes \mathcal{C} du décalage \mathbf{T} . Cet automate est réduit et synchronisé. Si le décalage \mathbf{T} est irréductible, la couverture de Krieger de \mathbf{T} a un unique sous-automate minimal et irréductible qui est la couverture de Shannon du décalage \mathbf{T} .*

Démonstration. Soit $\mathcal{K} = (V, A, \Delta)$ la couverture de Krieger d'un décalage d'arbres sofique \mathbf{T} et $\mathcal{C} = (V', A, \Delta')$ son automate des contextes. Soit t un arbre infini sur l'alphabet A et u_n son sous-arbre de hauteur n enraciné en ε . Ainsi pour tout entier $i \geq 0$, on a $\text{cont}_{\mathbf{T}}(u_{i+1}) \subseteq \text{cont}_{\mathbf{T}}(u_i)$. Comme le nombre de contextes du décalage \mathbf{T} est fini, il existe un entier strictement positif n tel que $\text{cont}_{\mathbf{T}}(u_{n+i}) = \text{cont}_{\mathbf{T}}(u_n)$ pour tout entier positif i . On définit alors $s(t) = \text{cont}_{\mathbf{T}}(u_n)$.

Montrons alors que la fonction de V dans V' qui à un contexte $\text{cont}_{\mathbf{T}}(t)$ associe $s(t)$ est bien définie, et qu'elle est de plus injective. Tout d'abord il est clair que $\text{cont}_{\mathbf{T}}(t) \subseteq s(t)$. Soit $c \in s(t) - \text{cont}_{\mathbf{T}}(t)$ un contexte dont la feuille marquée est en position x . Il existe un entier strictement positif n tel que $c(u_i) \in \mathcal{L}(\mathbf{T})$ pour tout entier $i \geq n$. Soit s un arbre infini tel que c est le sous-arbre de s enraciné en ε et t est le sous-arbre de s enraciné en x . Alors $s \in \mathbf{T}$ et $c \in \text{cont}_{\mathbf{T}}(t)$, donc $\text{cont}_{\mathbf{T}}(c) = s(t)$. En conséquence de quoi, si t et t' sont deux arbres infinis, alors $\text{cont}_{\mathbf{T}}(t) = \text{cont}_{\mathbf{T}}(t')$ si et seulement si $s(t) = s(t')$. Donc la fonction est bien définie et injective, et la couverture de Krieger est un sous-automate de l'automate des contextes.

Montrons à présent que l'automate \mathcal{K} est réduit et synchronisé. Si p est un état de \mathcal{K} égal au contexte $\text{cont}_{\mathbf{T}}(t)$ d'un arbre t sur l'alphabet A , alors $\text{cont}_{\mathcal{K}}(p) = \text{cont}_{\mathbf{T}}(t)$. L'automate \mathcal{K} est donc réduit. Soit $r_{\mathcal{A}}(t)$ le nombre d'états de l'automate \mathcal{A} sur lesquels un calcul de l'automate \mathcal{A} sur l'arbre t peut terminer. Par la Proposition 4.1.6, l'automate des contextes \mathcal{C} est synchronisé. Donc il existe un motif u tel que $r_{\mathcal{C}}(\text{cont}_{\mathbf{T}}(u)) = 1$. Il existe donc un arbre $t \in \mathbf{T}$ dont u est le sous-arbre enraciné en ε . Ainsi tous les calculs de l'automate des contextes \mathcal{C} sur l'arbre t terminent sur l'état $\text{cont}_{\mathbf{T}}(u)$. Comme

\mathcal{K} est un sous-automate de \mathcal{C} , tous les calculs de \mathcal{K} sur t terminent sur l'état $\text{cont}_{\mathbf{T}}(u)$. L'automate \mathcal{K} est donc synchronisé.

Supposons maintenant que le décalage \mathbf{T} est en plus irréductible. Soit z un bloc synchronisant de l'automate \mathcal{K} tel que tout calcul fini de \mathcal{K} sur z termine sur l'état p . Soit W l'ensemble des états accessibles depuis l'état p . Montrons que l'automate $\mathcal{K}' = (W, A, \Delta)$ est un sous-automate minimal de \mathcal{K} . Pour tout état $q \in W$, il existe un hyper-chemin de p vers q étiqueté par u . De plus, si on appelle v l'arbre fini obtenu en remplaçant chaque feuille de u par z , tout calcul de l'automate \mathcal{K} sur v termine dans l'état q . Comme \mathbf{T} est irréductible, il existe un arbre t dans \mathbf{T} et un code préfixe fini complet $P \subset \{0, 1\}^{\geq \text{hauteur}(z)}$, tel que z est le sous-arbre de t enraciné en ε , et v est le sous-arbre de t enraciné en tout $x \in P$. Il existe donc un hyper-chemin de q vers p dans l'automate \mathcal{K} . De plus, si r est un état d'un sous-automate minimal et irréductible de l'automate \mathcal{K} , sur lequel termine un calcul fini de \mathcal{K} sur un motif w , par irréductibilité du décalage \mathbf{T} , il existe un arbre t' dans \mathbf{T} et un code préfixe fini complet $Q \subset \{0, 1\}^{\geq \text{hauteur}(z)}$, tels que z est le sous-arbre de t' enraciné en ε , et que w est le sous-arbre de t' enraciné en tout $x \in Q$. Donc il existe un hyper-chemin de r vers p , et $p \in W$. Ainsi l'automate \mathcal{K}' est l'unique sous-automate minimal irréductible de l'automate \mathcal{K} , et c'est la couverture de Shannon du décalage \mathbf{T} . \square

4.1.5 Théorème de décomposition

Le théorème de décomposition pour les décalages sur \mathbb{Z} nous apprend que toute conjugaison entre deux décalages de type fini [Wil73] (ou sofiques [LM95, Exercice 7.1.10 p. 225]) peut être décomposée en une suite finie de conjugaisons élémentaires, les éclatements et les fusions. Le point clé de la démonstration de ce résultat est la possibilité de réduire la mémoire d'une fonction de bloc grâce à des éclatements d'états. Dans cette partie, nous montrons un théorème de décomposition pour les décalages d'arbres qui étend le théorème de décomposition pour les décalages d'arbres de type fini présenté dans [AB09].

Fonctions d'éclatement et de fusion

Soit \mathbf{T} un décalage d'arbres sur l'alphabet A . Pour toute lettre $a \in A$, on considère une partition \mathcal{P}_a des blocs du langage d'ordre 2, $\mathcal{L}_2(\mathbf{T})$, du décalage \mathbf{T} qui sont étiquetés par la lettre a à la racine. On notera par $[(a, b, c)]_a$ l'élément de la partition \mathcal{P}_a qui contient le bloc (a, b, c) ; il s'agit donc d'une classe d'équivalence.

Soit $\Phi : \mathbf{T} \rightarrow \mathcal{P}(\{0, 1\}^*)$ la fonction 2-bloc définie par $\phi(a, b, c) = [(a, b, c)]_a$.

On note par $\widetilde{\mathbf{T}}$ le décalage d'arbres $\Phi(\mathbf{T})$. La fonction Φ est une 2-conjugaison dont l'inverse est une fonction 1-bloc. On dit que Φ est une *fonction d'éclatement entrant* et que $\widetilde{\mathbf{T}}$, et plus généralement tout décalage d'arbres obtenu en renommant les lettres de $\widetilde{\mathbf{T}}$, est un *éclatement entrant* de \mathbf{T} . On dit aussi que Φ^{-1} est une *fonction de fusion entrante* et que \mathbf{T} est une *fusion entrante* de $\widetilde{\mathbf{T}}$. Quand \mathcal{P}_a est la partition triviale pour toute lettre a (i.e. $[(a, b, c)]_a = (a, b, c)$ pour toute lettre a), Φ est appelée *fonction d'éclatement entrant complète* et $\widetilde{\mathbf{T}}$ est l'*éclatement entrant complet* de \mathbf{T} .

Exemple 4.1.5. Soit l'alphabet à deux éléments $A = \{\bullet, \circ\}$, et on choisit comme partitions \mathcal{P}_\bullet et \mathcal{P}_\circ la partition triviale. On définit ainsi Φ la fonction d'éclatement entrant complète sur l'alphabet A .

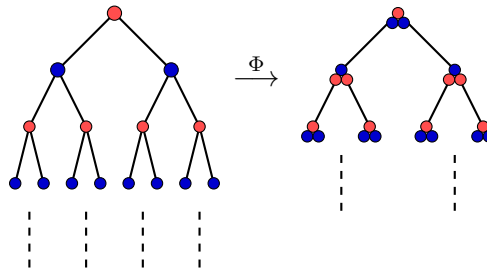
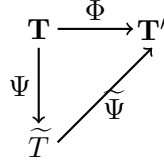


FIGURE 29: Exemple de fonction d'éclatement entrant complète Φ sur l'alphabet $A = \{\bullet, \circ\}$.

Énoncé et preuve du théorème

Théorème 4.1.10 ([AB09]). *Toute conjugaison entre deux décalages d'arbres peut être décomposée en une suite finie de fonctions d'éclatement entrants et de fonctions de fusions entrantes.*

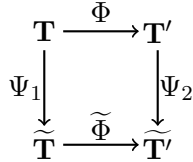
Lemme 4.1.11. *Soit $\Phi : \mathbf{T} \rightarrow \mathbf{T}'$ une m -conjugaison entre deux décalages d'arbres \mathbf{T} et \mathbf{T}' , avec $m \geq 2$. Alors il existe une fonction d'éclatement entrant Ψ de \mathbf{T} vers $\widetilde{\mathbf{T}}$ et une $(m-1)$ -conjugaison $\widetilde{\Phi}$ de $\widetilde{\mathbf{T}}$ vers \mathbf{T}' tels que $\Phi = \widetilde{\Phi} \circ \Psi$.*



Démonstration. On appelle A l'alphabet du décalage d'arbres \mathbf{T} et A' l'alphabet du décalage d'arbres \mathbf{T}' . On appelle $\varphi : \mathcal{L}_m(\mathbf{T}) \rightarrow A'$ la fonction locale qui définit la m -conjugaison Φ . Soit $\tilde{A} = \mathcal{L}_2(\mathbf{T})$ l'alphabet du décalage d'arbre \tilde{X} , et soit $\Psi_1 : \mathbf{T} \rightarrow \tilde{\mathbf{T}}$ la 2-conjugaison définie par $\psi_1(a, b, c) = (a, b, c)$. La fonction Ψ_1 est donc une fonction d'éclatement entrant complète.

Soit f (resp. g, h) la fonction de $\mathcal{L}_2(\mathbf{T})$ dans A qui envoie (a, b, c) sur a (resp. b, c). On définit $\tilde{\Phi} : \tilde{\mathbf{T}} \rightarrow \mathbf{T}'$ comme la $(m-1)$ -fonction de bloc telle que, pour tout bloc b de $\mathcal{L}_{m-1}(\tilde{X})$, par $\tilde{\varphi}(b) = \varphi(b')$, où b' est le bloc de hauteur m tel que $b'_x = f(b_x)$ pour tout $x \in \{0, 1\}^{<m-1}$, tel que $b'_{x_0} = g(b_x)$, et $b'_{x_1} = h(b_x)$ pour tout $x \in \{0, 1\}^{m-1}$. Ainsi on a $\Phi = \tilde{\Phi} \circ \Psi_1$. \square

Lemme 4.1.12. *Soit $\Phi : \mathbf{T} \rightarrow \mathbf{T}'$ une 1-conjugaison entre deux décalages d'arbres, telle que Φ^{-1} est une fonction m -bloc avec $m \geq 2$. Alors il existe une fonction d'éclatement entrant Ψ_1 de \mathbf{T} vers $\tilde{\mathbf{T}}$, une fonction d'éclatement entrant Ψ_2 de \mathbf{T}' vers $\tilde{\mathbf{T}}'$, et une 1-conjugaison $\tilde{\Phi}$ de $\tilde{\mathbf{T}}$ vers $\tilde{\mathbf{T}}'$ telle que $\Phi = \Psi_2^{-1} \circ \tilde{\Phi} \circ \Psi_1$ et $\tilde{\Phi}^{-1}$ est une $(m-1)$ -fonction de bloc. Le diagramme suivant est donc commutatif.*



Démonstration. Supposons que le décalage d'arbres \mathbf{T} (resp. \mathbf{T}') est défini sur l'alphabet A (resp. A'). Soit \mathcal{P}_a la partition des blocs de hauteurs 2 étiquetés par une lettre a à la racine, tels que deux blocs (a, b, c) et (a, b', c') appartiennent à la même classe de la partition si et seulement si $\phi(b) = \phi(b')$ et $\phi(c) = \phi(c')$, où $\phi : A \rightarrow A'$ est la fonction locale qui définit Φ . Soit $\Psi_1 : \mathbf{T} \rightarrow \tilde{\mathbf{T}}$ la fonction de bloc définie localement par $\psi_1(a, b, c) = [(a, b, c)]_a$. Alors $\tilde{\mathbf{T}}$ est un éclatement entrant de \mathbf{T} . Soit $\tilde{\mathbf{T}}'$ l'éclatement entrant complet de \mathbf{T}' sur

l'alphabet $A'_2 = \mathcal{L}_2(\mathbf{T}')$. On note Ψ_2 la fonction d'éclatement entrant complète de \mathbf{T}' vers $\widetilde{\mathbf{T}}$.

On définit une 1-fonction de bloc $\widetilde{\Phi}$ de $\widetilde{\mathbf{T}}$ vers $\widetilde{\mathbf{T}'}$, définie par la fonction locale $\widetilde{\phi}[(a, b, c)]_a = (\phi(a), \phi(b), \phi(c))$. Cette fonction locale est bien définie, de par le choix de la partition \mathcal{P}_a ; ainsi $\Phi = \Psi_2^{-1} \circ \widetilde{\Phi} \circ \Psi_1$. Il reste à vérifier que $\widetilde{\Phi}^{-1} = \Psi_2^{-1} \circ \Phi^{-1} \circ \Psi_1$ est une $(m-1)$ -fonction de bloc, autrement dit que pour tout arbre $t \in \widetilde{\mathbf{T}'}$, le bloc fini b de hauteur $m-1$ enraciné en ε dans l'arbre t détermine complètement $\widetilde{\Phi}^{-1}(t)_\varepsilon$. C'est effectivement le cas, car le bloc de hauteur $m-1$ enraciné en ε dans l'arbre t détermine tous les $\Psi_2^{-1}(t)_{x0}$ et tous les $\Psi_2^{-1}(t)_{x1}$, pour toute position $x \in \{0, 1\}^{m-1}$, et il détermine donc le bloc de hauteur m enraciné en ε de l'arbre $\Psi_2^{-1}(t)$. Ainsi si $t' = (\Phi^{-1} \circ \Psi_2^{-1})(t)$, la lettre t'_ε est entièrement déterminée par le bloc b . De plus le bloc de hauteur $m-1$ enraciné en ε dans l'arbre t détermine aussi $\Psi_1(t'_\varepsilon)$. \square

Nous avons à présent tous les éléments pour démontrer le Théorème 4.1.10.

Démonstration du Théorème 4.1.10. Soit $\Phi : \mathbf{T} \rightarrow \mathbf{T}'$ une n -conjugaison entre deux décalages d'arbres, telle que Φ^{-1} est une m -fonction de bloc, avec $n, m \geq 1$. Par le Lemme 4.1.11 et le Lemme 4.1.12, il existe des fonctions d'éclatements entrants $\Psi_1, \dots, \Psi_{n+m-2}, \Delta_1, \dots, \Delta_{m-1}$, et un renommage de l'alphabet Δ tels que $\Phi = \Delta_1^{-1} \circ \Delta_2^{-1} \cdots \circ \Delta_{m-1}^{-1} \circ \Delta^{-1} \circ \Psi_{n+m-2} \circ \Psi_2 \circ \Psi_1$. Un renommage étant un cas particulier d'éclatement entrant, le théorème est démontré. \square

4.2 Décalages de type fini

4.2.1 Décalages de sommets et de transitions

Dans cette partie nous présentons un cas particulier de décalages d'arbre de type fini : les décalages de sommets (*vertex shift* dans [LM95]).

Soit \mathbf{T} un décalage d'arbres défini sur un alphabet A . On appelle *décalage des blocs d'ordre n* du décalage \mathbf{T} le décalage de type fini $\mathbf{T}^{(n)}$ défini sur l'alphabet $A^{\mathbb{S}_n}$, et qui est l'image de \mathbf{T} par la n -fonction de bloc Φ_n définie localement par $\phi(p) = p$ pour tout motif $p \in \mathcal{L}_n(\mathbf{T})$. La fonction Φ_n est en fait la composée de $n-1$ fonctions d'éclatements entrants complets successifs, et donc tout décalage \mathbf{T} est conjugué à ses décalages des blocs d'ordre supérieur $\mathbf{T}^{(n)}$.

Un *décalage de sommets* est un décalage d'arbres accepté par un automate d'arbres $\mathcal{A} = (V, V, \Delta)$, où les transitions sont de la forme $(q_0, q_1, q) \rightarrow q$. Un arbre est accepté par un tel automate si chaque nœud est étiqueté par l'état correspondant dans le calcul de l'automate. L'ensemble des calculs d'un automate d'arbres est un décalage de sommets. Afin de simplifier les notations,

nous dirons qu'un décalage de sommets est l'ensemble des calculs d'un automate sans étiquette $B = (V, \Gamma)$ dont les transitions sont des éléments de $V^2 \times V$, et sont notées par $(q_0, q_1) \rightarrow q$. Cela revient à dire qu'un décalage de sommets est accepté par un automate d'arbres sans étiquette. En particulier un décalage d'arbres de sommets est un décalage d'arbres de type fini, car un automate d'arbres sans étiquette est 1-local.

Remarque. Un décalage irréductible de type fini n'est pas toujours conjugué à l'ensemble des calculs de son automate minimal. Par exemple le décalage trivial sur un alphabet à deux lettres est accepté par un automate à un seul état.

Proposition 4.2.1. *Tout décalage d'arbres de type fini est conjugué à un décalage d'arbres de sommets.*

Démonstration. Soit $\mathbf{T} = \mathbf{T}_{\mathbb{F}}$ un décalage d'arbres de type fini, défini par l'ensemble fini de blocs interdits F tous de hauteur m , avec $m \geq 1$. Soit $\mathcal{A} = (V, A, \delta)$ l'automate d'arbres déterministe m -local dont l'ensemble d'états est $V = \mathcal{L}_m(\mathbf{T})$. Pour $p_0, p_1 \in V$ et $a \in A$, on note $\text{tronc}(b)$ le bloc de hauteur $m - 1$ tel que $\text{tronc}(b)_x = b_x$ pour tout $x \in \{0, 1\}^{\leq m-1}$, où b est le bloc (a, p_0, p_1) . On définit alors $\delta(p_0, p_1, a) = \text{tronc}(b)$ si et seulement si b est un bloc autorisé de \mathbf{T} .

L'automate d'arbre \mathcal{A} accepte \mathbf{T} , et pour tout arbre $t \in \mathbf{T}$, il existe un unique calcul de \mathcal{A} sur t . Soit \mathbf{T}' le décalage de sommets composé de tous les calculs de l'automate \mathcal{A} . On considère $\mathbf{T}^{(m)}$ le décalage des blocs d'ordre m de \mathbf{T} . Alors $\mathbf{T}^{(m)}$ est exactement \mathbf{T}' , et donc \mathbf{T} est conjugué à \mathbf{T}' . \square

Un décalage d'arbres de transitions est un décalage d'arbres accepté par un automate dont toutes les transitions ont des étiquettes différentes. Un décalage d'arbres de transitions est donc un décalage d'arbres de type fini.

Proposition 4.2.2. *Tout décalage d'arbres de type fini est conjugué à un décalage d'arbres de transitions.*

Démonstration. Si \mathbf{T} est un décalage de type fini d'ordre n , alors sa représentation d'ordre n est un décalage de transitions. \square

4.2.2 Conjugaison des décalages de type fini

Dans cette partie nous montrons, grâce au théorème de décomposition, que le problème de la conjugaison des décalages d'arbres de type fini est décidable. Pour cela nous commençons par montrer que deux fusions entrantes commutent.

Proposition 4.2.3. *Supposons que \mathbf{T}_1 est un décalage de sommets, et que \mathbf{T}_2 , \mathbf{T}_3 sont des décalages de sommets obtenus par fusion entrante de \mathbf{T}_1 . Alors il existe un décalage de sommets \mathbf{T}_4 que l'on peut obtenir à la fois comme fusion entrante de \mathbf{T}_2 et de \mathbf{T}_3 .*

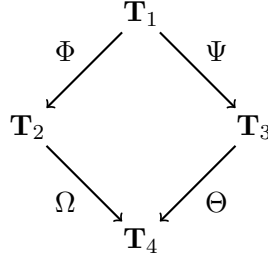


FIGURE 30: Les fusions entrantes commutent. Si les décalages de sommets \mathbf{T}_2 et \mathbf{T}_3 sont des fusions entrantes de \mathbf{T}_1 , il existe un décalage d'arbres de sommets \mathbf{T}_4 qui est une fusion entrante commune à \mathbf{T}_2 et \mathbf{T}_3 .

Dans la Figure 30, les fonctions Φ et Ψ sont des fusions entrantes. Il s'ensuit par la Proposition 4.2.3 que les fonctions Ω et Θ sont aussi des fusions entrantes.

Il est possible de traduire les fonctions d'éclatements entrants et de fusions sortantes sur un décalage de sommets par une transformation sur un automate d'arbres sans étiquette qui reconnaît ce-dernier.

Soit \mathbf{T} un décalage de sommets accepté par un automate d'arbres sans étiquette $\mathcal{A} = (V, \Delta)$, et Φ une fonction d'éclatement entrant de \mathbf{T} vers $\tilde{\mathbf{T}}$ définie par des partitions des ensembles de transitions Δ_r partant de l'état r pour tout sommet $r \in V$. On note par \tilde{V} l'alphabet du décalage $\tilde{\mathbf{T}}$. Il s'agit de l'union des ensembles $\{[(r, p, q)]_r, \mid (p, q) \rightarrow r \in \Delta\}$. Les sommets $[(r, p, q)]_r$ sont appelés *sommets éclatés* du sommet r . On dit que les sommets $[(r, p, q)]_r$ sont *fusionnés* en le sommet r . Le décalage de sommets $\tilde{\mathbf{T}}$ est accepté par l'automate $\tilde{\mathcal{A}} = (\tilde{V}, \tilde{\Delta})$, défini par $([(r, p, q)]_r, [(r', p', q')]_{r'}) \rightarrow [(s, r, r')]_s \in \tilde{\Delta}$ si et seulement si $(r, r') \rightarrow s \in \Delta$.

Pour plus de lisibilité, les sommets de $\tilde{\Delta}$ correspondants à une partition de l'ensemble Δ_r sont notés $r^1 \dots, r^{\ell(r)}$. Une fusion entrante d'un décalage de sommets \mathbf{T} accepté par un automate \mathcal{A} est un décalage de sommets $\tilde{\mathbf{T}}$ accepté par un automate $\tilde{\mathcal{A}}$ tel que $\tilde{\mathbf{T}}$ est un éclatement entrant de \mathbf{T} . Les sommets $r^1 \dots, r^{\ell(r)}$ de l'automate $\tilde{\mathcal{A}}$ sont fusionnés en le sommet r . Remarquons que lorsque $(p', q') \rightarrow r^i \in \tilde{\Delta}$, alors $(p', q') \rightarrow r^j \notin \tilde{\Delta}$, pour tous $p', q' \in \tilde{V}$ et tous $1 \leq i \neq j \leq \ell(r)$. Ceci implique en particulier que $(r^i, p') \rightarrow q' \in \tilde{\Delta}$ si

et seulement si $(r^j, p') \rightarrow q' \in \widetilde{\Delta}$, et $(p', r^i) \rightarrow q' \in \widetilde{\Delta}$ pour toute transition $(p', p^j) \rightarrow q' \in \widetilde{\Delta}$, pour tous sommets $p', q' \in \widetilde{V}$ et pour tous $1 \leq i, j \leq \ell(r)$. De manière informelle, si un sommet r est éclaté en sommets $r^1 \dots, r^{\ell(r)}$, les transitions entrantes sur r sont réparties entre les sommets r^i , alors que les transitions sortantes de r sont dupliquées.

Démonstration. Soit $\mathcal{A}_i = (V_i, \Delta_i)$ un automate d'arbres sans étiquette qui reconnaît le décalage de sommets \mathbf{T}_i pour $i = 1, 2$ et 3 . Supposons tout d'abord qu'il existe une fusion entrante $\Phi : \mathbf{T}_1 \rightarrow \mathbf{T}_2$ et une fusion entrante $\Psi : \mathbf{T}_1 \rightarrow \mathbf{T}_3$. Les états $p^1, \dots, p^{l(p)}$ de V_1 sont fusionnés en un état p de V_2 . Par définition d'une fusion entrante, cela implique que si $(q, r) \rightarrow p^i \in \Delta_1$, alors $(q, r) \rightarrow p^j \notin \Delta_1$ pour tous états $q, r \in V_1$ et tous $1 \leq i \neq j \leq l(p)$. Cela implique aussi que $(p^i, q) \rightarrow r \in \Delta_1$ si et seulement si $(p^j, q) \rightarrow r \in \Delta_1$, et $(q, p^i) \rightarrow r \in \Delta_1$ si et seulement si $(q, p^j) \rightarrow r \in \Delta_1$ pour tous états $q, r \in V_1$ et tous $1 \leq i, j \leq l(p)$. Nous supposons aussi que par ailleurs les états $q^1, \dots, q^{l(q)}$ de V_1 sont fusionnés en un état q de V_3 .

Le cas le plus simple est celui où les états $p^1, \dots, p^{l(p)}$ et $q^1, \dots, q^{l(q)}$ sont tous distincts. Il suffit de définir le décalage \mathbf{T}_4 comme la fusion entrante de \mathbf{T}_2 obtenue en fusionnant les états $p, q^1, \dots, q^{l(q)}$ en un état q . Il s'agit aussi de la fusion entrante de \mathbf{T}_3 obtenue en fusionnant les états $q, p^1, \dots, p^{l(p)}$ en un état q . Supposons à présent que $p^1 = q^1, \dots, p^l = q^l$ pour un entier $1 \leq l \leq \min(l(p), l(q))$. Cela implique que, pour tous $1 \leq i \leq l(p)$, $1 \leq j \leq l(q)$, on a $(p^i, q) \rightarrow r \in \Delta_n$ si et seulement si $(p^j, q) \rightarrow r \in \Delta_n$, et $(q, p^i) \rightarrow r \in \Delta_n$ si et seulement si $(q, p^j) \rightarrow r \in \Delta_n$ pour $n = 1$ et $n = 2$. Nous définissons le décalage \mathbf{T}_4 comme la fusion entrante de \mathbf{T}_2 obtenue en fusionnant les états $p, q^{l+1}, \dots, q^{l(q)}$ en un état p . Il s'agit aussi de la fusion entrante de \mathbf{T}_3 obtenue en fusionnant les états $q, p^{l+1}, \dots, p^{l(p)}$ en un état p . Ainsi, si les fonctions Φ et Ψ sont des fusions entrantes, il en est de même des fonctions Ω et Θ . \square

Grâce à la Proposition 4.2.3, il nous est possible de définir la *fusion minimale* d'un décalage sofique \mathbf{T} , comme le décalage d'arbres de sommets défini par l'automate d'arbres $\mathcal{A} = (V, \Delta)$ avec le plus petit nombre de sommets, cet automate \mathcal{A} étant obtenu par des fusions entrantes successives appliquées à un automate reconnaissant le décalage \mathbf{T} . Cette fusion minimale est bien unique, car si un décalage \mathbf{T} possédait deux fusions minimales \mathbf{T}_1 et \mathbf{T}_2 , la Proposition 4.2.3 nous permet de construire une fusion entrante commune \mathbf{T}_3 , qui par minimalité de \mathbf{T}_1 et \mathbf{T}_2 est telle que $\mathbf{T}_1 = \mathbf{T}_2 = \mathbf{T}_3$.

Théorème 4.2.4. *Soient \mathbf{T}_1 et \mathbf{T}_2 deux décalages d'arbres de type fini. Alors savoir si \mathbf{T}_1 et \mathbf{T}_2 sont conjugués est un problème décidable.*

Démonstration. La Proposition 4.2.1 permet de supposer que les décalages \mathbf{T}_1 et \mathbf{T}_2 sont des décalages de sommets. Par le Théorème 4.1.10, il existe une suite de fonctions d'éclatements entrants et de fonctions de fusions entrantes transformant \mathbf{T}_1 en \mathbf{T}_2 .

Supposons dans un premier temps que cette suite se décompose en une suite de fonctions d'éclatements entrants de \mathbf{T}_1 vers \mathbf{T} , suivie (à un renommage près des sommets de \mathbf{T}) d'une suite de fonctions de fusions entrantes de \mathbf{T} vers \mathbf{T}_2 . Ce cas particulier est illustré dans la Figure 31. La Proposition 4.2.3 nous assure l'existence d'un décalage de sommets à la confluence de deux flèches en pointillé dans la Figure 31. Ainsi les décalages \mathbf{T}_1 et \mathbf{T}_2 ont une fusion commune, et donc la même fusion minimale. Réciproquement si \mathbf{T}_1 et \mathbf{T}_2 ont la même fusion minimale, il existe une suite de fonctions d'éclatements entrants et de fonctions de fusions entrantes de \mathbf{T}_1 vers \mathbf{T}_2 .

Supposons maintenant qu'il existe une suite quelconque d'éclatements entrants et de fusions entrantes de \mathbf{T}_1 vers \mathbf{T}_2 . Cette suite peut se décomposer en plusieurs suites de la forme précédente, et nous arrivons à la même conclusion par transitivité. \square

Le schéma de la preuve nous donne un algorithme pour calculer, si elle existe, une conjugaison entre deux décalages d'arbres de type fini \mathbf{T}_1 et \mathbf{T}_2 . La Proposition 4.2.1 nous permet de nous ramener à des décalages de sommets : $\mathbf{Y}_1 = \psi_1(\mathbf{T}_1)$ et $\mathbf{Y}_2 = \psi_2(\mathbf{T}_2)$, où ψ_i est une suite de fonctions d'éclatements entrants et \mathbf{Y}_i un décalage d'arbres de sommets. Par la Proposition 4.2.3, on se ramène à leur fusion minimale : $\phi_1(\mathbf{Y}_1) = \tilde{\mathbf{Y}}_1$ et $\phi_2(\mathbf{Y}_2) = \tilde{\mathbf{Y}}_2$ où ψ_i est une suite de fonctions de fusions entrantes et $\tilde{\mathbf{Y}}_i$ est une fusion minimale. Il nous suffit alors de comparer les $\tilde{\mathbf{Y}}_i$: s'ils sont différents alors les décalages \mathbf{T}_1 et \mathbf{T}_2 ne sont pas conjugués, et s'ils sont égaux nous avons une conjugaison $\mathbf{T}_1 = \psi_1^{-1} \circ \phi_1^{-1} \circ \phi_2 \circ \psi_2(\mathbf{T}_2)$. Cet algorithme fonctionne en temps exponentiel, car la comparaison des fusions minimales $\tilde{\mathbf{Y}}_1$ et $\tilde{\mathbf{Y}}_2$ revient à un problème d'isomorphisme de graphes.

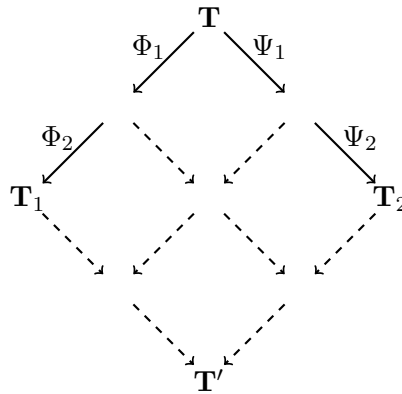


FIGURE 31: Une suite de fonctions d'éclatements entrants de \mathbf{T}_1 vers \mathbf{T} suivie (à un renommage près de l'alphabet de \mathbf{T}), par une suite de fonctions de fusions entrantes de \mathbf{T} vers \mathbf{T}_2 . Chaque flèche représente une fonction de fusion entrante. Les décalages d'arbres \mathbf{T}_1 et \mathbf{T}_2 ont la même fusion minimale \mathbf{T}' .

4.3 Les décalages presque de type fini (AFT)

La notion de décalage d'arbres presque de type fini étend celle connue pour les décalage sur \mathbb{Z} , initialement introduite dans [Mar85]. Pour les décalages sur \mathbb{Z} , elle constitue une classe intéressante notamment en théorie des codes [MK88]. Cette classe de décalages contient la classe des SFT irréductibles, et est strictement incluse dans la classe des décalages sofiques irréductibles.

4.3.1 Définition

Soient \mathbf{T}, \mathbf{T}' deux décalages d'arbre. Une fonction de bloc $\Phi : X \rightarrow Y$ est *fermante à gauche* s'il existe deux entiers positifs m, a (appelés mémoire et anticipation) tels que, si $\Phi(s) = s'$ et $\Phi(t) = t'$ avec $s'_x = t'_x$ pour tout $x \in \{0, 1\}^i$ avec $0 \leq i \leq (a + m)$, et $s_x = t_x$ pour tout $x \in \{0, 1\}^i$ avec $0 \leq i \leq m - 1$, alors $s_x = t_x$ pour tout $x \in \{0, 1\}^m$.

Un automate d'arbres $\mathcal{A} = (V, A, \Delta)$ est *fermant à gauche* si deux calculs de l'automate \mathcal{A} sur un même arbre qui terminent dans un même état sont les mêmes. Cela revient à dire qu'il existe un entier positif a tel que deux calculs finis C, C' de l'automate \mathcal{A} sur un même bloc $u \in \mathcal{L}_a(\mathbf{T})$ tels que $C_\varepsilon = C'_\varepsilon$ vérifient $C_x = C'_x$ pour $x \in \{0, 1\}$. La notion d'automate d'arbres fermant à

gauche correspond à celle d'automate co-déterministe avec délai fini pour les mots (voir par exemple [Sak09]).

Une fonction de bloc $\Phi : \mathbf{T} \rightarrow \mathbf{T}'$ est *fermante à droite* s'il existe des entiers positifs m, a (appelés mémoire et anticipation) tels que, lorsque $\Phi(s) = s'$, $\Phi(t) = t'$ avec $s'_x = t'_x$ pour tout $x \in \{0, 1\}^i$ avec $0 \leq i \leq (a + m)$, et $s_x = t_x$ pour tout $x \in \{0, 1\}^i$ avec $a + 1 \leq i \leq (a + m)$, alors $s_x = t_x$ pour tout $x \in \{0, 1\}^a$. La fonction Φ est *résoluble à droite* si c'est une fonction 1-bloc fermante à droite avec paramètres $m = 1$ et $a = 0$.

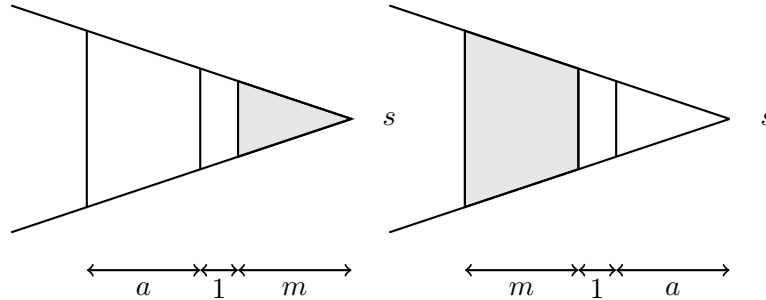


FIGURE 32: Les notions d'automate fermant à gauche et à droite. Les arbres sont ici dessinés horizontalement pour mettre en évidence la provenance des deux notions.

Soit Φ une 1-fonction de bloc de \mathbf{T} dans \mathbf{T}' , définie par une fonction locale ϕ . On dit que le bloc z est *résoluble pour Φ* s'il existe un entier m tel que, si u et v sont deux blocs de hauteur m qui apparaissent dans \mathbf{T} qui vérifient $\phi(u) = \phi(v)$, alors $u_\varepsilon = v_\varepsilon$. Cette notion correspond à celle de *resolving block* définie dans [Mar85].

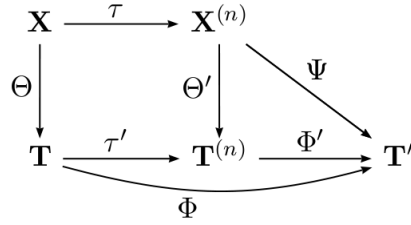
Définition 16. Un décalage d'arbres sofique est *presque de type fini* (on parlera dans la suite de décalage AFT, de l'anglais *almost of finite type*) si c'est l'image d'un décalage d'arbres irréductible de type fini par une fonction de bloc qui est à la fois résoluble à droite, fermante à gauche, et qui possède un bloc résoluble.

Il est facile de vérifier que la composée d'une conjugaison avec une fonction de bloc fermante à gauche est encore fermante à gauche. De même, une conjugaison préserve la propriété de posséder un bloc résoluble pour une fonction de bloc.

La proposition suivante montre que le caractère AFT d'un décalage d'arbres est un invariant de conjugaison.

Proposition 4.3.1 ([AB10]). *Soient \mathbf{T} et \mathbf{T}' deux décalages d'arbres irréductible conjugués. Alors \mathbf{T} est AFT si et seulement si \mathbf{T}' est AFT.*

Démonstration. Soit \mathbf{T} un décalage d'arbres AFT. Soit Θ une 1-fonction de bloc résoluble à droite, fermante à gauche et qui possède un bloc résoluble, d'un décalage de type fini \mathbf{X} vers \mathbf{T} . Soit Φ une conjugaison entre \mathbf{T} et \mathbf{T}' . On suppose que Φ est une m -conjugaison dont l'inverse est une n -conjugaison. Soit $\mathbf{X}^{(n)}$ (resp. $\mathbf{T}^{(n)}$) le décalage des blocs d'ordre n de \mathbf{X} (resp. \mathbf{T}), et τ (resp. τ') la conjugaison naturelle entre \mathbf{X} et $\mathbf{X}^{(n)}$ (resp. entre \mathbf{T} et $\mathbf{T}^{(n)}$). Le décalage d'arbres $\mathbf{X}^{(n)}$ est de type fini. On peut étendre la fonction de bloc Θ en une 1-fonction de bloc Θ' de $\mathbf{X}^{(n)}$ vers $\mathbf{T}^{(n)}$, de sorte que Θ' est elle aussi une 1-fonction de bloc résoluble à droite, fermante à gauche et avec un bloc résoluble. Soit $\Phi' = \Phi \circ \tau'^{(-1)}$, qui est donc une conjugaison, et $\Psi = \Phi' \circ \Theta'$. La fonction de bloc Ψ est une fonction 1-bloc résoluble à droite. Elle est aussi fermante à gauche et possède aussi un bloc résoluble, donc le décalage d'arbres \mathbf{T}' est AFT. \square



On dit qu'un automate d'arbres est *presque de type fini* (on dira aussi que l'automate d'arbres est AFT) s'il est déterministe, irréductible, fermant à gauche et synchronisé.

Proposition 4.3.2 ([AB10]). *Un décalage d'arbres est AFT si et seulement si il est accepté par un automate d'arbres AFT.*

Démonstration. Soit \mathbf{T} un décalage d'arbres AFT sur un alphabet A . Soit $\Phi : \mathbf{X} \rightarrow \mathbf{T}$ une fonction de bloc surjective entre un décalage d'arbres de type fini irréductible \mathbf{X} et le décalage \mathbf{T} , qui est donc résoluble à droite, fermante à gauche et qui a un bloc résoluble.

Sans perte de généralité (en remplaçant \mathbf{X} par un de ses décalages des blocs d'ordre supérieur), on peut supposer que \mathbf{X} est un décalage de transitions et que Φ est une fonction 1-bloc. Encore une fois en remplaçant le décalage \mathbf{X} par une de ses décalage des blocs d'ordre supérieur, on peut suppose que Φ est fermante à gauche avec paramètres $a' = 0, m' = 1$.

Soit $\mathcal{A} = (V, E, \Delta)$ un automate d'arbres irréductible et de transitions qui accepte le décalage \mathbf{X} . Soit $\mathcal{A}' = (V, A, \Delta')$ l'automate dont les transitions sont

$(p, q, \phi(e), r) \in \Delta'$ si et seulement si $(p, q, e, r) \in \Delta$. Comme Φ est une fonction 1-bloc résoluble à droite, \mathcal{A}' est un automate déterministe.

On montre que l'automate d'arbres \mathcal{A}' possède un bloc synchronisant. Comme Φ possède un bloc résoluble z , c'est un bloc synchronisant pour \mathcal{A}' . On émonde l'automate \mathcal{A}' en ne gardant que les états accessibles depuis p_z . Comme le décalage \mathbf{T} est irréductible, l'automate \mathcal{A}' reste irréductible et accepte toujours \mathbf{T} .

Montrons à présent que l'automate d'arbres \mathcal{A}' est fermant à gauche. Si ce n'était pas le cas, il existerait deux calculs distincts C et C' de l'automate \mathcal{A}' sur un même arbre t terminant dans un même état p . Soit $(p, q, e, r) \in \Delta$ une transition sortant du couple d'états (p, q) pour des états $p, r \in V$ (s'il n'existe pas de transition de ce type, alors comme l'automate \mathcal{A} est irréductible il existe une transition sortant du couple d'états (q, p) et la suite du raisonnement est la même). On obtient ainsi deux calculs distincts (r, C, D) et (r, C', D) de l'automate \mathcal{A}' sur le même arbre $u = (\phi(e), t, t')$ et qui terminent dans l'état r . Ces deux calculs distincts de \mathcal{A}' sont aussi des calculs distincts de l'automate \mathcal{A} sur deux arbres s, s' du décalage \mathbf{T} donnés par $s_\varepsilon = s'_\varepsilon = e$ et $\Phi(s) = \Phi(s')$. Comme Φ est fermante à gauche avec paramètres $a' = 0, m' = 1$, on en déduit que $s = s'$ et donc $C = C'$.

Réciproquement, on suppose que \mathbf{T} est accepté par un automate d'arbres AFT $\mathcal{A} = (V, A, \Delta)$. Soit \mathbf{X} le décalage de sommets accepté par $\mathcal{A}' = (V, \Delta, \Delta')$, dont les transitions sont de la forme $(p, q, (p, q, a, r)) \rightarrow r$ pour $(p, q, a, r) \in \Delta$. Soit Φ la 1-fonction de bloc de \mathbf{X} sur \mathbf{T} définie par $\phi(p, q, a, r) = a$. Cette fonction de bloc est résoluble à droite puisque l'automate \mathcal{A} est déterministe. Elle est fermante à gauche car \mathcal{A} l'est. Enfin, comme l'automate \mathcal{A} est synchronisé, la fonction Φ possède un bloc résoluble. Ceci montre que le décalage \mathbf{T} est AFT. \square

4.3.2 Caractérisation des AFT par leur couverture de Shannon

Corollaire 4.3.3 ([AB10]). *Un décalage d'arbres sofique irréductible est AFT si et seulement si sa couverture de Shannon est AFT.*

Démonstration. Soit \mathbf{T} un décalage d'arbres irréductible et sofique. Par la Proposition 4.1.8, tout décalage sofique irréductible est accepté par un automate d'arbres irréductible, déterministe et synchronisé \mathcal{S} , qui est en fait la couverture de Shannon du décalage.

Supposons que le décalage \mathbf{T} est AFT. Par la Proposition 4.3.2, on sait que \mathcal{S} est égal à la minimisation d'un automate AFT \mathcal{A} . Montrons que \mathcal{S} est fermant à gauche. Si ce n'est pas le cas, il existe un arbre $t \in \mathbf{T}$ et deux calculs distincts

de \mathcal{S} sur t terminant dans le même état. En conséquence, il existe deux calculs différents de \mathcal{A} sur l'arbre t qui terminent dans deux états p, p' ayant le même contexte.

Soit z un bloc synchronisant pour l'automate \mathcal{A} , qui synchronise sur l'état q_z . Comme \mathcal{A} est irréductible, il existe un hyper-chemin de p vers q étiqueté par un motif w , et tel que z est enraciné en ε . Soit P le code préfixe fini complet qui définit cet hyper-chemin. Alors il existe un calcul C de l'automate \mathcal{A} sur un arbre s tel que, pour tout nœud $x \in P$, l'arbre t est enraciné en x dans s , $C_x = p$ et le motif z est enraciné en ε dans s . Comme p et p' ont même contexte, il existe aussi un calcul C' de l'automate \mathcal{A} sur l'arbre s tel que, pour un certain nœud $x \in P$, on a $C'_x = p'$. Comme le bloc z est synchronisant, on en déduit que $C_\varepsilon = C'_\varepsilon$. On obtient ainsi deux calculs distincts de l'automate \mathcal{A} sur un même arbre s terminant tous deux dans l'état q , ce qui contredit le caractère fermant à gauche de \mathcal{A} . Ceci montre que la couverture de Shannon d'un décalage AFT est fermante à gauche.

Réciproquement supposons que \mathcal{S} est AFT. Alors par la Proposition 4.3.2 le décalage \mathbf{T} est aussi AFT. \square

4.4 Procédures de décision

4.4.1 Calcul de la couverture de Shannon

Soit $\mathcal{A} = (V, A, \delta)$ un automate d'arbres. On rappelle que $\mathcal{D}(\mathcal{A})$ désigne son automate déterminisé.

Proposition 4.4.1. *Il est possible de vérifier en temps polynomial si un automate d'arbres est irréductible.*

Démonstration. Soit $\mathcal{A} = (V, A, \delta)$ un automate d'arbres déterministe. Pour tout état $p \in V$, on définit l'ensemble des P_i , pour $i \geq 0$, par induction de la manière suivante :

- $P_0 = \{p\}$;
- $P_1 = \{q \in V \mid \exists a \in A \text{ tel que } \delta(p, p, a) = q\}$;
- pour tout entier n strictement positif,

$$P_{n+1} = \{q \in V \mid \exists a \in A, p_1, p_2 \in P_n \cup \{p\} \text{ tels que } \delta(p_1, p_2, a) = q\} \cup P_n.$$

La suite d'ensembles de sommets $(P_n)_{n \in \mathbb{N}}$ est par définition une suite croissante, et bornée par l'ensemble fini V . Il existe donc un entier n_0 tel que pour tout

$n \geq n_0$, on a $P_n = P_{n_0}$. Par construction, il existe un hyper-chemin de p à q si et seulement si $q \in P_{n_0}$, et donc l'automate \mathcal{A} est irréductible si et seulement si pour tout état p , l'ensemble P_{n_0} défini précédemment coïncide avec l'ensemble des états V . Cet algorithme est polynomial en le nombre d'états $|V|$ de l'automate car pour chaque état p , le calcul de P_{n_0} s'effectue en au plus $|V|$ étapes, et pour un état p donné le calcul de chaque P_i est polynomial en $|V|$. \square

Proposition 4.4.2. *On peut décider si un automate d'arbres est synchronisé.*

Démonstration. Soit \mathcal{A} un automate d'arbres $\mathcal{D}(A)$ son déterminisé. Alors l'automate \mathcal{A} est synchronisé si et seulement si l'ensemble des états de son déterminisé $\mathcal{D}(A)$ contient un singleton. La complexité en temps et en espace de cet algorithme est exponentiel en le nombre d'états $|V|$ de l'automate \mathcal{A} . \square

Proposition 4.4.3. *Soit $\mathcal{A} = (V, A, \delta)$ un automate d'arbres déterministe qui accepte un décalage sofique irréductible \mathbf{T} . Alors la couverture de Shannon de \mathbf{T} est calculable à partir de \mathcal{A} .*

Démonstration. Soit $\mathcal{D}(\mathcal{A}) = (\mathfrak{P}(V), A, \delta')$ et R un état minimal de $\mathcal{D}(\mathcal{A})$ au sens de l'inclusion. Soit u l'étiquette d'un hyper-chemin entre V et R . Alors u est un motif synchronisant pour $\mathcal{D}(\mathcal{A})$. En effet par minimalité de R , tout calcul fini de $\mathcal{D}(\mathcal{A})$ sur u termine dans l'état R . Comme \mathbf{T} est irréductible, en ne conservant dans $\mathcal{D}(\mathcal{A})$ que les états accessibles depuis R on obtient un automate d'arbres déterministe, irréductible et synchronisé qui accepte le décalage \mathbf{T} . De la Proposition 4.1.8 on déduit que cette opération de réduction nous donne la couverture de Shannon de \mathbf{T} . \square

4.4.2 Automate des paires et graphe des paires

Sur des mots finis, l'automate des paires est utilisé pour vérifier des propriétés sur les paires de chemins de l'automate (voir [Sak09, p. 647]). Nous étendons cette notion d'automate des paires, ainsi que celle de graphes des paires, aux arbres afin de vérifier certaines propriétés des automates d'arbre.

Étant donné un automate d'arbres $\mathcal{A} = (V, A, \Delta)$, on définit l'automate des paires de \mathcal{A} , que l'on notera $\mathcal{A} \times \mathcal{A} = ((V^2 \times V^2) \cup V^2, A, \Delta')$, comme l'automate déterministe dont les transitions sont $((p, p'), (q, q')), a \rightarrow (r, r')$ si et seulement si $(p, q), a \rightarrow r$ et $(p', q'), a \rightarrow r'$ sont des transitions de \mathcal{A} . Un état diagonal de l'automate des paires $\mathcal{A} \times \mathcal{A}$ est un état (p, p) pour $p \in V$, ou un état $((p, p), (q, q))$ pour $p, q \in V$.

Une représentation de l'automate des paires est donnée par le graphe des paires. Soit $\mathcal{A} = (V, A, \Delta)$ un automate d'arbres. Le graphe des paires $G_{\mathcal{A}} =$

(V_G, E_G) de l'automate \mathcal{A} , où $V_G \subseteq (V^2 \times V^2) \cup V^2$ est l'ensemble des sommets et $E_G \subseteq V_G \times \{0, 1\} \times A \times V_G$ est l'ensemble des arêtes. L'ensemble E_G est défini de la manière suivante (pour plus de lisibilité, une arête étiquetée par 1 sera représentée par un trait plein \longrightarrow , et est appelée arête pleine, tandis qu'une arête étiquetée par 0 est représentée par un trait en pointillé \dashrightarrow , et est appelée arête pointillée). Pour toute paire de transitions étiquetées par la même lettre $(p, q), a \rightarrow r$ et $(p', q'), a \rightarrow r'$ et pour toute paire (s, s') d'états de l'automate \mathcal{A} , l'ensemble d'arêtes E_G contient les arêtes suivantes :

$$\begin{aligned} ((p, p'), (q, q')) &\dashrightarrow^a ((r, r'), (s, s')), \\ ((p, p'), (q, q')) &\xrightarrow{a} ((s, s'), (r, r')), \\ ((p, p'), (q, q')) &\dashrightarrow^a (r, r'), \\ ((p, p'), (q, q')) &\xrightarrow{a} (r, r'). \end{aligned}$$

Les étiquettes par des lettres de l'alphabet A portées par les arêtes de G peuvent être retirées afin de réduire la complexité du graphe.

Un sommet du graphe $G_{\mathcal{A}}$ est *utile* s'il possède au moins une arête pleine entrante et une arête pointillée entrante. On ne conserve que la *partie essentielle* du graphe des paires, que l'on obtient en supprimant les sommets qui ne sont pas utiles, ainsi que les arêtes arrivant sur ces sommets ou partant de ces sommets.

Proposition 4.4.4. *On peut calculer la partie essentielle du graphe des paires d'un automate d'arbres \mathcal{A} en temps linéaire en $\mathcal{O}(|E_G| + |V_G|)$.*

Démonstration. On suppose que l'automate d'arbres $\mathcal{A} = (A, V, \delta)$ est déterministe. Le cardinal de $|V_G|$ est alors un $\mathcal{O}(|V|^4)$ et le cardinal de $|E_G|$ est un $\mathcal{O}(|V|^6)$. On appelle *0-prédécesseur* un prédécesseur d'un sommet par une flèche pointillée (d'étiquette 0), et *1-prédécesseur* un prédécesseur d'un sommet par une flèche pleine (d'étiquette 1). On définit de façon similaire les notions de 0-successeur et de 1-successeur. On construit une file d'attente *nodeQueue* de sommets qui doivent être retirés du graphe des paires. Dès qu'un état est retiré de la file d'attente, on retire les arêtes sortant de cet état.

PARTIE ESSENTIELLE(graphe des paires $G_{\mathcal{A}}$)

```

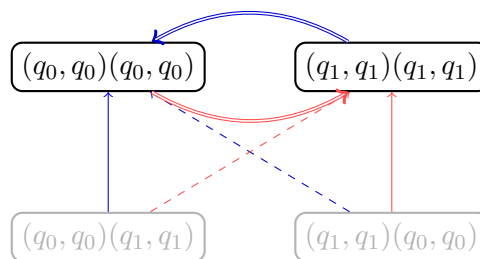
1  Soit  $n_0(u)$  le nombre de 0-prédécesseurs du sommets  $u \in V_G$ .
2  Soit  $n_1(u)$  le nombre de 1-prédécesseurs du sommets  $u \in V_G$ .
3   $vertexQueue \leftarrow$  liste des sommets  $u$  tels que  $n_0(u) = 0$  ou  $n_1(u) = 0$ .
4  Marquer les sommets contenus dans  $vertexQueue$ 
5  while  $vertexQueue$  est non-vide
6      do retirer le sommet  $u$  de  $vertexQueue$ 
7          for tout 0-successeur  $v$  de  $u$ 
8              do diminuer  $n_0(v)$ 
9                  if  $n_0(v) = 0$  et  $v$  n'est pas marqué
10                     then ajouter  $v$  à  $vertexQueue$ 
11                         marquer  $v$ 
12          for tout 1-successeur  $v$  de  $u$ 
13              do diminuer  $n_1(v)$ 
14                  if  $n_1(v) = 0$  et  $v$  n'est pas marqué
15                     then ajouter  $v$  à  $vertexQueue$ 
16                         marquer  $v$ 
17  return les états non marqués

```

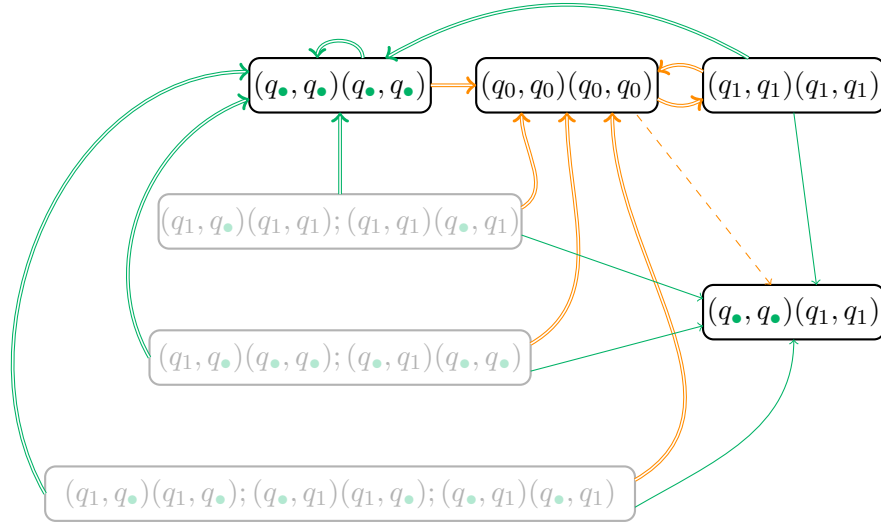
□

Un sommet $((p, q), (r, s))$ (resp. (p, q)) du graphe des paires G est *non diagonal* si $p \neq q$ ou bien $r \neq s$ (resp. $p \neq q$). Dans la suite on parlera du graphe des paires pour désigner sa partie essentielle.

Exemple 4.4.1. Le graphe des paires de l'automate d'arbres de l'Exemple 4.1.1 est représenté ci-dessous. Les sommets utiles sont cerclés de noirs, les autres de gris. Les couleurs des flèches correspondent à la lettre qu'elle porte comme étiquette. Une double flèche remplace une flèche pleine et une flèche pointillée portant la même étiquette. Pour plus de lisibilité, le sommet (p, q) est confondu avec le sommet $((p, q), (p, q))$.



Exemple 4.4.2. Le graphe des paires de l'automate d'arbres de l'Exemple 4.1.2 est représenté ci-dessous, avec les mêmes conventions que pour l'exemple précédent.



La proposition suivante, qui découle directement de la définition du graphe des paires, énonce un résultat qui sera utilisé pour montrer que le graphe des paires permet de décider certaines propriétés sur les décalages sofiques, comme la localité (voir la Proposition 4.4.7) ou la fermeture à gauche (voir la Proposition 4.4.9).

Proposition 4.4.5. Soit $\mathcal{A} = (V, A, \Delta)$ un automate d'arbres. Un sommet $((p, p'), (q, q'))$ est un sommet du graphe des paires $G_{\mathcal{A}}$ (en ne conservant que sa partie essentielle) si et seulement s'il existe un arbre s avec deux calculs de \mathcal{A} sur s , l'un terminant sur p et l'autre sur p' , ainsi qu'un arbre t avec deux calculs de \mathcal{A} sur t , l'un terminant sur q et l'autre sur q' .

Démonstration. Ceci provient du fait qu'on n'a conservé que la partie essentielle du graphe des paires, et que donc les sommets non utiles ont été supprimés. De plus, s'il existe une arête $((r, r'), (s, s')) \xrightarrow{0,a} ((p, p'), (q, q'))$ (ou une arête $(r, r') \xrightarrow{0,a} ((p, p'), (q, q'))$) dans le graphe des paires $G_{\mathcal{A}}$, alors il existe deux calculs E et E' de l'automate \mathcal{A} tels que $\sigma^0(E) = C$, $\sigma^0(E') = C'$, $\sigma^1(E) = D$, $\sigma^1(E') = D'$ (σ étant l'action naturelle par translation de $\{0, 1\}^*$ sur les arbres), le calcul E termine dans l'état r et le calcul E' termine dans l'état r' . \square

Remarque. Il existe une arête $((r, r'), (s, s')) \xrightarrow{0,a} ((p, p'), (q, q'))$ (ou une arête $(r, r') \xrightarrow{0,a} ((p, p'), (q, q'))$) dans le graphe des paires $G_{\mathcal{A}}$ si et seulement s'il existe une lettre $a \in A$ et deux transitions $(p, q) \xrightarrow{a} r$ et $(p', q') \xrightarrow{a} r'$ (ou $(p, q) \xrightarrow{a} s$ et $(p', q') \xrightarrow{a} s'$) dans l'automate \mathcal{A}

4.4.3 Localité

Proposition 4.4.6. *Un automate d'arbres est local si et seulement s'il existe un calcul dans son automate des paires qui termine sur un état non diagonal.*

Démonstration. Par définition de l'automate des paires $\mathcal{A} \times \mathcal{A}$, l'existence dans $\mathcal{A} \times \mathcal{A}$ d'un calcul se terminant sur un état (p, q) avec $p \neq q$ implique l'existence dans \mathcal{A} de deux calculs distincts sur un même arbre. Réciproquement, s'il existe deux calculs distincts de \mathcal{A} sur un même arbre t , ces deux calculs diffèrent en un certain nœud $x \in \{0, 1\}^*$. D'où l'existence de deux calculs sur le sous-arbre de t en position x qui se terminent dans deux états distincts. \square

Proposition 4.4.7. *Un automate d'arbres \mathcal{A} n'est pas local si et seulement si la partie essentielle de son graphe des paires contient un état non diagonal (p, q) avec $p \neq q$.*

Démonstration. Soit $G_{\mathcal{A}}$ le graphe des paires d'un automate d'arbres \mathcal{A} . Si \mathcal{A} n'est pas local, alors il existe deux calculs de l'automate \mathcal{A} sur un même arbre qui terminent dans les états r et r' respectivement, avec $r \neq r'$. Par la Proposition 4.4.5, c'est équivalent au fait que le sommet (r, r') soit dans la partie essentielle du graphe des paires.

Réciproquement, si $G_{\mathcal{A}}$ contient un sommet non diagonal $((r, r'), (s, s'))$, avec $r \neq r'$ et $s \neq s'$ (ou (r, r') avec $r \neq r'$), alors il existe une lettre $a \in A$ et une arête $((p, p'), (q, q')) \xrightarrow{0,a} ((r, r'), (s, s'))$ ou une arête $((p, p'), (q, q')) \xrightarrow{0,a} (r, r')$. Par la Proposition 4.4.5 et la remarque qui la suit, il existe deux calculs de \mathcal{A} sur un même arbre t , l'un terminant sur l'état r et l'autre terminant sur l'état r' . \square

Pour un automate d'arbres déterministe $\mathcal{A} = (A, V, \delta)$, le graphe des paires $G_{\mathcal{A}}$ possède $\mathcal{O}(|V|^4)$ sommets et $\mathcal{O}(|V|^6)$ arêtes. Il est donc possible de vérifier en temps $\mathcal{O}(|V|^6)$ si un automate d'arbres est local.

Exemple 4.4.3. L'automate d'arbres \mathcal{A}_1 de l'Exemple 4.1.1 est local car la partie essentielle de son graphe des paires ne contient que des états diagonaux. L'automate d'arbres \mathcal{A}_1 de l'Exemple 4.1.2 n'est pas local car la partie essentielle de son graphe des paires contient l'état non diagonal $(q_{\bullet}, q_{\bullet})(q_1, q_1)$.

4.4.4 Automate fermant à gauche

Proposition 4.4.8. *Un automate d'arbres n'est pas fermant à gauche si et seulement s'il existe un calcul de son automate des paires terminant dans un état diagonal et passant par un état non diagonal.*

Démonstration. Soit \mathcal{A} un automate d'arbres déterministe. Par définition de l'automate des paires, l'existence d'un calcul dans $\mathcal{A} \times \mathcal{A}$ terminant dans un état diagonal (p, p) et passant par un état (r, s) avec $r \neq s$ est équivalent à l'existence de deux calculs distincts de l'automate \mathcal{A} sur un même arbre et qui terminent sur un même sommet. \square

Proposition 4.4.9. *Un automate d'arbres n'est pas fermant à gauche si et seulement s'il existe un chemin dans son graphe des paires (dont on n'a conservé que la partie essentielle) partant d'un sommet non diagonal et arrivant sur un sommet (p, p) .*

Démonstration. Soit $G_{\mathcal{A}}$ le graphe des paires d'un automate d'arbres \mathcal{A} . Si \mathcal{A} n'est pas fermant à gauche, alors il existe deux calculs distincts C et C' de l'automate sur un même arbre t terminant dans un même état p . Soit x un nœud de l'arbre t tel que $C_x \neq C'_x$. Alors il existe dans le graphe des paires $G_{\mathcal{A}}$ un chemin étiqueté par (x, w) partant d'un sommet $((C_x, C'_x), (s, s'))$ ou un sommet $((s, s'), (C_x, C'_x))$ (selon la dernière lettre 0 ou 1 du nœud x) et terminant sur un sommet (p, p) , où w est l'étiquette du chemin de la racine de l'arbre t au nœud x .

Réciproquement, supposons qu'il existe un chemin dans le graphe des paires $G_{\mathcal{A}}$ entre un sommet $((p, p'), (q, q'))$, avec $p \neq p'$ ou $q \neq q'$, et un sommet (r, r) . Cela signifie qu'il existe un arbre t et deux calculs C et C' de l'automate \mathcal{A} sur cet arbre t qui terminent sur l'état p , et tels qu'il existe des nœuds $x, y \in \{0, 1\}^*$ avec $C_x = p$, $C'_x = q$ et $C_y = q$, $C'_y = q'$. Cela implique que les deux calculs C et C' sont distincts, et donc l'automate \mathcal{A} n'est pas fermant à gauche. \square

Proposition 4.4.10. *Soit \mathbf{T} un décalage d'arbres irréductible accepté par un automate d'arbre \mathcal{A} . On peut décider si \mathbf{T} est un AFT.*

Démonstration. On commence par calculer la couverture de Shannon \mathcal{S} du décalage d'arbre \mathbf{T} , comme indiqué dans la Proposition 4.4.3. Il suffit ensuite de vérifier qu'elle est AFT. Cela revient à vérifier qu'elle est fermante à gauche, car par définition la couverture de Shannon d'un décalage d'arbres est au automate d'arbres déterministe, irréductible et synchronisé, ce qui est possible d'après la Proposition 4.4.9. Le nombre de sommets du graphe des paires $G_{\mathcal{S}}$ est un

$\mathcal{O}(|V|^4)$ et son nombre d'arêtes est un $\mathcal{O}(|V|^6)$. La propriété de la Proposition 4.4.9 peut être vérifiée en temps linéaire en la taille du graphe des paires G_S . On peut donc vérifier en temps polynomial si la couverture de Shannon d'un décalage d'arbres sofique irréductible est AFT. \square

Remarque. L'algorithme proposé par Seidl [Sei89] pour vérifier que le degré d'ambiguïté d'un automate d'arbres est fini possède une complexité du même ordre (cube du nombre de transitions de l'automate).

Conclusion et perspectives

Conclusion

Dans cette thèse nous avons présenté des résultats concernant d'une part les décalages sur \mathbb{Z}^d , et d'autre part les décalages d'arbres.

Décalages sur \mathbb{Z}^d Le principal résultat concernant les décalages sur \mathbb{Z}^d est le Théorème 3.1.1, qui établit que tout décalage effectif de dimension d peut être vu comme une sous-action projective d'un décalage sofique de dimension $d + 1$. Ce résultat, outre le fait qu'il améliore celui d'Hochman [Hoc09], montre que les décalages de type fini sur \mathbb{Z}^d contiennent en un certain sens tous les décalages calculables. Il existe certainement d'autres applications de ce résultat que celles présentées dans la Partie 3.2. Ce théorème de simulation permet de mieux comprendre l'opération de sous-action projective, qui correspond en terme de systèmes dynamiques à étudier des sous-sytèmes. Il serait intéressant de trouver d'autres groupes que \mathbb{Z}^d sur lesquels un théorème du même type serait valable.

Décalages d'arbres Il ressort des résultats obtenus pour les décalages d'arbres une forte similarité entre ceux-ci et les décalages sur \mathbb{N} . Cette similarité est due à deux principales raisons. Tout d'abord sur \mathbb{M}_2 on peut définir des automates qui possèdent quasiment toutes les bonnes propriétés des automates sur les mots. Nous avons vu qu'il était possible d'adapter des notions spécifiques à la dimension 1, comme par exemple l'irréductibilité, aux décalages d'arbres (voir la Partie 4.1.2). Ensuite la structure unidirectionnelle du monoïde \mathbb{M}_2 donne un rôle particulier au point origine des configurations, et permet par exemple de rendre la conjugaison des décalages de type fini décidable. La motivation initiale pour étudier ces décalages d'arbres était d'essayer de comprendre en quoi définir des décalages sur \mathbb{N} ou \mathbb{N}^d modifient leurs propriétés, mais au vu de cette étude, il apparaît que les décalages d'arbres sont peut-être trop proches de ceux sur \mathbb{N} . Il serait intéressant de s'éloigner un peu plus de la dimension 1, en considérant par exemple des décalages définis sur des monoïdes virtuellement libres (qui possèdent un sous-monoïde libre d'indice fini).

Décalages sur un monoïde finiment présenté Le Chapitre 1 présentait la notion de décalage sur un monoïde finiment présenté, et définit trois classes de décalages : décalages de type fini, décalages sofiqes et décalages effectifs. Mais on sait peu de choses concernant ces classes lorsque les décalages sont définis sur des monoïdes ou groupes qui ne sont pas \mathbb{N}^d , \mathbb{Z}^d ou \mathbb{M}_d ; il est donc naturel de se demander lesquels des résultats connus sur ces structures s'adaptent au cas général. Plus précisément, on peut chercher à caractériser, pour chaque propriété, les monoïdes qui la vérifient. Une telle démarche a déjà été réalisée pour la dynamique symbolique sur des groupes hyperboliques [CP93], ou dans le cas des automates cellulaires sur des groupes [Fio00, CSC09]. Nous présentons maintenant quelques pistes qui permettraient de poursuivre les recherches initiées dans cette thèse.

Théorème de décomposition général

Dans le Chapitre 4 nous avons défini, pour les décalages d'arbres, des automates d'arbres qui nous ont permis de mieux comprendre la classe des décalages d'arbres sofiqes. Comme cela a été précisé au début de ce chapitre, il est facile d'obtenir les mêmes résultats pour les décalages sofiqes sur le monoïde libre à n générateurs \mathbb{M}_n (au lieu de deux générateurs pour les arbres).

Les décalages ont été définis au Chapitre 1 sur des monoïdes finiment présentés. Est-il possible de généraliser, comme cela a été fait pour les décalages sur le monoïde libre \mathbb{M}_n , la notion d'automate à ces structures ? Des automates fini reconnaissant des motifs finis ou des configurations infinies ont déjà été étudiés sur un graphe de Cayley d'un monoïde ou d'un groupe finiment présenté [Gar93]. Il est certainement possible d'adapter ces définitions pour faire en sorte que les décalages sofiqes sur un monoïde finiment présenté soient exactement ceux reconnus par de tels automates.

Question 1. Comment définir des automates reconnaissant les décalages sofiqes sur un monoïde finiment présenté ?

Le théorème de décomposition, énoncé dans la Partie 4.1.5 pour des décalages d'arbres quelconques, est basé sur des fonctions de bloc élémentaires, les fusions et les éclatements. Dans le cas des arbres, et de manière plus générale sur n'importe quel monoïde libre, il suffit de définir des fusions entrantes et des éclatements entrants. Pour généraliser le théorème de décomposition à un groupe, il faudrait aussi y faire intervenir des fusions sortantes et des éclatements sortants, définis de manière analogue aux transformations entrantes, en remplaçant les générateurs du groupe par leur inverse.

Conjecture 1. *On peut écrire un théorème de décomposition pour des conjugaisons entre décalages sur un monoïde finiment présenté.*

La question qui suit naturellement l'énoncé du théorème de décomposition est celle de la décidabilité du problème de conjugaison.

Question 2. A quelle condition sur le monoïde \mathbb{M} la conjugaison des décalages de type fini est-elle un problème décidable ?

Les travaux de Muller et Schupp [MS85] d'une part et de Kuske et Lohrey [KL05] d'autre part donnent des pistes dans cette direction. Les auteurs donnent en particulier une condition suffisante sur un groupe finiment présenté \mathbb{G} pour que le problème du domino (savoir si un décalage de type fini contient au moins une configuration) soit décidable. Lorsque le problème du domino est indécidable, il en est de même pour le problème de conjugaison.

Proposition ([MS85, KL05]). *Si \mathbb{G} est un groupe virtuellement libre, alors le problème du domino sur \mathbb{G} est un problème décidable.*

Point de vue algébrique

A tout décalage sofique de dimension 1, on peut associer son monoïde syntaxique, qui permet de caractériser les décalages sofiques : un décalage est sofique si et seulement si son monoïde syntaxique est fini [Sch56]. Diverses propriétés d'un décalage sofique peuvent être lues sur son monoïde syntaxique, et nous renvoyons à [Pin86] pour plus de détails.

Concernant les décalages d'arbres définis au Chapitre 4, on peut s'inspirer de travaux existants pour les arbres finis [BW06, Boj08, Boj09]. Ces travaux doivent cependant être adaptés, car les arbres qui y sont considérés sont d'une part finis, et d'autre part pas nécessairement des blocs.

Conjecture 2 ([AB11c]). *On peut définir une algèbre d'arbres qui permet de caractériser la classe décalages d'arbres de type fini et la classe des décalages d'arbres presque de type fini.*

Encore une fois, il serait intéressant de chercher à généraliser ce résultat à d'autres structures que les monoïdes libres.

Question 3. Comment définir une algèbre permettant de caractériser des classes ou propriétés de décalages sur un monoïde finiment présenté quelconque ?

Table des figures

1	Exemples de graphes de Cayley	12
2	Action de \mathbb{M}_2	14
3	Supports et motifs	16
4	Fonction de bloc	22
5	Un exemple de suite $(\mathbf{k}^n)_{n \in \mathbb{Z}}$ de vecteurs 2-dimensionnels qui déforme le réseau \mathbb{Z}^2	31
6	Une substitution agit sur un configuration.	32
7	Un ensemble de substitutions \mathcal{S} qui possède la propriété <i>A</i>	37
8	Les motifs autorisés du décalage de type fini \mathbf{T}	42
9	Exemple de sous-action sofique.	43
10	Exemple de calcul d'une machine de Turing.	48
11	Substitution s_1 pour définir des zones de calcul.	51
12	Trois itérations de la substitution s_1	52
13	Zones de calcul de différents niveaux.	52
14	Un canal de communication entre les tuiles de calcul i et f	54
15	Communication à l'intérieur d'une zone de calcul.	55
16	Une horloge sur les zones de calcul.	56
17	Calculs d'une machine de Turing dans des zones de différents niveaux.	60
18	La substitution s_2 qui définit les canaux de communication.	66
19	Rectangles de communication.	67
20	Les bandes de dépendance associée à des zones de calcul de niveau 2.	68
21	Adresses dans une bande de dépendance.	69
22	Zones de responsabilité.	70
23	Détection d'un motif interdit dans la zone de responsabilité.	72
24	Arbre de recherche d'une requête.	74
25	Décalage uniformément mélangeant.	81
26	Machine de Turing à semi-oracle codée dans un décalage sofique.	85
27	Notion de localité pour un automate d'arbre	97

28	Décalage d'arbres irréductible et automate d'arbres irréductible.	100
29	Exemple de fonction d'éclatement.	109
30	Commutation des fonctions de fusions entrantes.	113
31	Confluence des fonctions de fusions entrantes.	116
32	Automate d'arbres fermant à gauche et à droite.	117

Bibliographie

- [AB09] Nathalie AUBRUN et Marie-Pierre BÉAL : Decidability of conjugacy of tree-shifts of finite type. *In ICALP '09 : Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 132–143, Berlin, Heidelberg, 2009. Springer-Verlag.
- [AB10] Nathalie AUBRUN et Marie-Pierre BÉAL : Sofic and almost of finite type tree-shifts. *In CSR 2010 : 5th International Computer Science Symposium in Russia*, volume 6072 de *Lecture Notes in Computer Science*, pages 12–24. Springer, 2010.
- [AB11a] Nathalie AUBRUN et Marie-Pierre BÉAL : A decomposition theorem for tree-shifts. *Pré-publication*, 2011.
- [AB11b] Nathalie AUBRUN et Marie-Pierre BÉAL : On AFT tree-shifts. *Pré-publication*, 2011.
- [AB11c] Nathalie AUBRUN et Marie-Pierre BÉAL : Tree algebra of sofic tree languages. *Soumis*, 2011.
- [AS09] Nathalie AUBRUN et Mathieu SABLIK : An order on sets of tilings corresponding to an order on languages. *In Susanne ALBERS et Jean-Yves MARION, éditeurs : STACS 2009 : Proceedings of the 26th Annual Symposium on the Theoretical Aspects of Computer Science*, pages 99–110, Freiburg Allemagne, 2009. IBFI Schloss Dagstuhl.
- [AS10] Nathalie AUBRUN et Mathieu SABLIK : Simulation of effective subshifts by two-dimensional subshifts of finite type. *Soumis*, 2010.
- [AS11] Nathalie AUBRUN et Mathieu SABLIK : Multidimensional effective s-adic systems are sofic. *Soumis*, 2011.
- [BBrEP10] Marie-Pierre BÉAL, Jean BERSTEL, Søren EILERS et Dominique PERRIN : Symbolic dynamics. *CoRR*, abs/1006.1265, 2010.
- [Ber66] Robert BERGER : *The Undecidability of the Domino Problem*. American Mathematical Society, 1966.

- [BJ10] Alexis BALLIER et Emmanuel JEANDEL : Computing (or not) quasi-periodicity functions of tilings. *Second symposium on Cellular Automata (JAC 2010)*, 2010.
- [Boj08] Mikolaj BOJAŃCZYK : Effective characterizations of tree logics. *In Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '08, pages 53–66, New York, NY, USA, 2008. ACM.
- [Boj09] Mikolaj BOJANCZYK : Algebra for tree languages. *In CSL*, Lecture Notes in Computer Science, page 1. Springer, 2009.
- [BW06] Mikolaj BOJAŃCZYK et Igor WALUKIEWICZ : Characterizing ef and ex tree logics. *Theoretical Computer Science*, 358:255–272, August 2006.
- [CDG⁺07] Hubert COMON, Max DAUCHET, Rémi GILLERON, Christof LÖDING, Florent JACQUEMARD, Denis LUGIEZ, Sophie TISON et Marc TOMMASI : Tree automata techniques and applications, 2007.
- [CH] Julien CASSAIGNE et Michael HOCHMAN : non publié.
- [Col86] Donald COLLINS : A simple presentation of a group with unsolvable word problem. *Illinois Journal of Mathematics*, 30:230–234, 1986.
- [CP93] Michel COORNAERT et Athanase PAPADOPOULOS : *Symbolic dynamics and hyperbolic groups*. Springer, 1993.
- [CSC09] Tullio CECCHERINI-SILBERSTEIN et Michel COORNAERT : *Cellular Automata and Groups*. Springer, 2009.
- [Die05] Reinhard DIESTEL : *Graph Theory (Graduate Texts in Mathematics)*. Springer, 2005.
- [DLS01] Bruno DURAND, Leonid LEVIN et Alexander SHEN : Complex tilings. *In STOC '01 : Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 732–739, New York, NY, USA, 2001. ACM.
- [DRS10a] Bruno DURAND, Andrei ROMASHCHENKO et Alexander SHEN : 1D Effectively Closed Subshifts and 2D Tilings. *In Second Symposium on Cellular Automata (JAC)*, pages 2–7, 2010.

- [DRS10b] Bruno DURAND, Andrei ROMASHCHENKO et Alexander SHEN : Effective closed subshifts in 1d can be implemented in 2d. In Andreas BLASS, Nachum DERSHOWITZ et Wolfgang REISIG, éditeurs : *Fields of Logic and Computation*, volume 6300 de *Lecture Notes in Computer Science*, pages 208–226. Springer Berlin / Heidelberg, 2010.
- [Dur00] Fabien DURAND : Linearly recurrent subshifts have a finite number of non-periodic subshift factors. *Ergodic Theory and Dynamical Systems*, 20(4):1061–1078, 2000.
- [Fio00] Francesca FIORENZI : *Cellular automata and finitely generated groups*. Thèse de doctorat, University of Rome "La Sapienza", 2000.
- [Fis75] Roland FISCHER : Sofic systems and graphs. *Monatshefte für Mathematik*, 80:179–186, 1975.
- [Gar93] Max H. GARZON : Cayley automata. *Theoretical Computer Science*, 108(1):83–102, 1993.
- [GM07] Anahí GAJARDO et Jacques MAZOYER : One head machines from a symbolic approach. *Theoretical Computer Science*, 370:34–47, 2007.
- [GS98] Chaim GOODMAN-STRAUSS : Matching rules and substitution tilings. *Annals of Mathematics*, 157:181–223, 1998.
- [Hed69] Gustav Arnold HEDLUND : Endomorphisms and automorphisms of the shift dynamical system. *Mathematical Systems Theory*, 3:320–337, 1969.
- [HM38] Gustav HEDLUND et Marston MORSE : Symbolic dynamics. *American Journal of Mathematics*, 60(4):815–866, 1938.
- [HM10] Michael HOCHMAN et Tom MEYEROVITCH : A characterization of the entropies of multidimensional shifts of finite type. *Annals of Mathematics (2)*, 171(3):2011–2038, 2010.
- [Hoc09] Michael HOCHMAN : On the dynamics and recursive properties of multidimensional symbolic systems. *Inventiones Mathematicae*, 176(1):131–167, 2009.
- [Kit98] Bruce KITCHENS : *Symbolic Dynamics*. Springer New York, 1998.

- [KL05] Dietrich KUSKE et Markus LOHREY : Logical aspects of cayley-graphs : the group case. *Annals of Pure and Applied Logic*, 131(1-3):263–286, 2005.
- [LM95] Douglas LIND et Brian MARCUS : *An Introduction to Symbolic Dynamics and Coding*. Cambridge, 1995.
- [Mar85] Brian MARCUS : Sofic systems and encoding data. *IEEE Transactions on Information Theory*, 31:366–377, 1985.
- [MK88] Brian MARCUS et Razmik KARABED : Sliding-block coding for input-restricted channels. *IEEE Transactions on Information Theory*, 34:2–26, 1988.
- [Moz89] Shahar MOZES : Tilings, substitution systems and dynamical systems generated by them. *Journal d'Analyse Mathématique (Jerusalem)*, 53:139–186, 1989.
- [MS85] David E. MULLER et Paul E. SCHUPP : The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985.
- [Nas95] Masakazu NASU : Textile systems for endomorphisms and automorphisms of the shift. *Memoirs of the American Mathematical Society*, 546, 1995.
- [Pan84] Jean-Jacques PANSIOT : Complexité des facteurs des mots infinis engendrés par morphismes itérés. In *Automata, languages and programming (Antwerp, 1984)*, volume 172 de *Lecture Notes in Computer Science*, pages 380–389. Springer, 1984.
- [Pin86] Jean-Éric PIN : *Varieties of Formal Languages*. Foundations of Computer Science. Plenum Publishing Corp., New York, 1986.
- [PS10] Ronnie PAVLOV et Michael SCHRAUDNER : Classification of sofic projective subdynamics of multidimensional shifts of finite type. *Soumis*, 2010.
- [Rob71] Raphael ROBINSON : Undecidability and nonperiodicity for tilings of the plane. *Inventiones Mathematicae*, 12:177–209, 1971.
- [Rog87] Hartley ROGERS : *Theory of Recursive Functions and Effective Computability*. MIT Press Cambridge, MA, USA, 1987.

- [Rot94] Joseph ROTMAN : *An Introduction to the Theory of Groups*. Springer-Verlag, 1994.
- [Sak09] Jacques SAKAROVITCH : *Elements of Automata Theory*. Cambridge University Press, 2009.
- [Sch56] Marcel-Paul SCHÜTZENBERGER : Une théorie algébrique du codage. *Séminaire Dubreil. Algèbre et Théorie des Nombres*, 9:1–24, 1956.
- [Sei89] Helmut SEIDL : On the finite degree of ambiguity of finite tree automata. In *Fundamentals of computation theory (Szeged, 1989)*, volume 380 de *Lecture Notes in Computer Science*, pages 395–404. Springer, New York, 1989.
- [Wan61] Hao WANG : Proving theorems by pattern recognition ii. *Bell System Technical Journal*, 40(1-3):1–41, 1961.
- [Wei73] Benjamin WEISS : Subshifts of finite type and sofic systems. *Monatshefte für Mathematik*, 77:462–474, 1973.
- [Wil73] R. F. WILLIAMS : Classification of subshifts of finite type. *Annals of Mathematics*, 98:120–153 ; errata, *ibid.* 99 (1974), 380–381, 1973.