Automates infinis et traces de Mazurkiewicz

Alexandre Mansard

Thèse de Doctorat de l'Université de la Réunion

Spécialité

Mathématiques - Informatique

présentée par

Alexandre Mansard

pour obtenir le grade de

Docteur de l'Université de la Réunion

Automates infinis et traces de Mazurkiewicz

soutenue publiquement le 24 novembre 2020 au Laboratoire d'Informatique et de Mathématiques de l'Université de la Réunion devant le jury composé de :

Didier Caucal, Directeur de recherches CNRS (Co-directeur)

Victor Chepoi, Professeur (Examinateur)

Christian Delhommé, Professeur (Directeur)

Volker Diekert, Professeur (Rapporteur)

Kamal Lodaya, Professeur (Rapporteur)

Marianne Morillon, Professeur (Examinateur)

Chloé Rispal, Maitre de conférences (Examinateur)

Sophie Tison, Professeur (Examinateur)

Remerciements

Je remercie l'ensemble des membres du jury d'avoir accepté d'évaluer ce travail.

Je remercie sincèrement et chaleureusement Christian Delhommé et Didier Caucal d'avoir accepté de diriger mes travaux. Je les remercie plus particulièrement pour la confiance qu'ils m'ont témoignée, pour l'apport scientifique dont ils m'ont fait bénéficié, pour leur bienveillance, leurs encouragements permanents, leur dynamisme, leur disponibilité.

Je remercie ma belle-famille pour le soutien logistique et les délicieux petits repas.

Je ne saurais trouver les mots justes pour remercier mes parents et mon frère pour leur présence et leur réconfort, en toute circonstance et depuis toujours.

Enfin, je dédie ce modeste travail à mes trois amours, mon épouse Déborah et mes enfants Grégoire et Timothée.

Résumé

Nous introduisons la notion de régularité par niveaux pour des langages de traces de Mazurkiewicz et nous considérons des systèmes reconnaissables de réécriture de traces, à contextes réguliers par niveaux (RTL). Nous prouvons qu'un automate dont le graphe sous-jacent est le graphe de réécriture d'un système RTL et dont les ensembles de sommets initiaux et finaux sont réguliers par niveaux (automate RTL), est mot-automatique. En particulier, la théorie du premier ordre d'un automate RTL est décidable. Ensuite, nous prouvons que, enrichi de la relation d'accessibilité, un automate dont le graphe sous-jacent est déplié concurrent d'un graphe fini concurrent et dont les ensembles de sommets initiaux et finaux sont réguliers par niveaux, est RTL. En particulier, la théorie du premier ordre avec accessibilité d'un tel automate est décidable. Par ailleurs, il est bien connu que la théorie du premier ordre avec accessibilité du graphe de réécriture suffixe d'un système de réécriture de termes clos (graphe GTR) est décidable. Nous mettons en évidence divers dépliés concurrents de graphes finis concurrents qui ne sont pas des graphes GTR. L'arbre du quart de la grille infinie est un exemple de tel déplié. La classe des dépliés concurrents des graphes finis concurrents constitue ainsi une classe de DAG mot-automatiques, dont la théorie du premier ordre avec accessibilité est décidable et qui contient des graphes non GTR.

Nous définissons pour les automates de traces (automates dont les sommets sont des traces de Mazurkiewicz) deux opérations que sont la synchronisation par niveaux et la superposition par niveaux et nous montrons que si une famille $\mathcal{F}$ d'automates de traces est fermée pour ces opérations, alors pour tout automate déterministe $H \in \mathcal{F}$, les langages acceptés par les automates déterministes de $\mathcal{F}$ qui sont longueur-réductibles en H forment une algèbre de Boole ; la longueur d'une trace étant donnée par la longueur de sa forme normale de Foata, un automate de traces G est longueur-réductible dans un automate de traces H, s'il existe un morphisme de G dans H préservant la longueur. Ensuite, nous montrons que la classe des automates suffixes de traces à contextes réguliers par niveaux, qui n'est que l'extension aux traces de Mazurkiewicz des automates suffixes de mots, satisfait ces propriétés de fermeture. Nous appelons réseau de Petri généralisé un automate suffixe de traces sur un alphabet de dépendance pour lequel la dépendance est réduite à l'égalité. Nous montrons alors que la sous-famille des réseaux de Petri généralisés satisfait également les propriétés de fermeture ci-dessus. Cela conduit notamment à diverses algèbres de Boole de langages acceptés par des réseaux de Petri généralisés déterministes.

Abstract

We introduce the notion of level-regularity for Mazurkiewicz trace languages and we consider recognizable trace rewriting systems with level-regular contexts (RTL system). We prove that an automaton for which the underlying graph is the rewriting graph of a RTL system and for which the sets of initial vertices and final vertices are level-regular (RTL automaton), is word-automatic. In particular, the first-order theory of a RTL automaton is decidable. Then, we prove that, enriched with the reachability relation, an automaton for which the underlying graph is the concurrent unfolding of a finite concurrent graph, and for which the sets of initial vertices and final vertices are level-regular, is RTL. In particular, the first-order theory with the reachability predicate of such an automaton is decidable. Besides, it is known that this property also holds for ground term rewriting graphs (GTR graph). We highlight various concurrent unfoldings of finite concurrent graphs that are not GTR graphs. The infinite quarter grid tree is such an unfolding. The class of concurrent unfoldings of finite concurrent graphs is therefore a class of word-automatic graphs for which the first-order theory with the reachability predicate is decidable and that contains some non GTR graphs.

We define the operations of level-length synchronization and level-length superposition of trace automata (automata for which vertices are Mazurkiewicz traces) and we prove that if a family $\mathcal{F}$ of trace automata is closed under these operations, then for any deterministic trace automaton $H \in \mathcal{F}$, the languages accepted by the deterministic trace automata belonging to $\mathcal{F}$ and that are length-reducible to H, form a Boolean algebra ; the length of a trace being the length of its Foata normal form, a trace automaton G is length-reducible to a trace automaton H if there exists a length-preserving morphism from G to H. Then, we show that the family of trace suffix automata with level regular contexts, the extension of word suffix automata to Mazurkiewicz traces, satisfies these closure properties. We define a generalized Petri net as a trace suffix automaton over a dependence alphabet for which the dependence is reduced to the equality and we show that the subfamily of generalized Petri nets also satisfies the closure properties above. In particular, this yields various Boolean algebras of word languages accepted by deterministic generalized Petri nets.

Table des matières

Chapitre 1

Introduction

Ce travail s'inscrit dans l'étude structurelle des automates infinis de présentation finie [46]. Cette notion d'automate infini est le résultat notamment de la mise en rapport des notions de langages, de graphes infinis de présentation finie et de logiques, que nous allons rappeler.

Langage

Les langages réguliers, algébriques, contextuels et récursivement énumérables forment une hiérarchie croissante de familles de langages, définie par Chomsky à partir de grammaires de complexité croissante [13]. Les propriétés de fermeture de ces langages ont été étudiées. Par exemple, la classe des langages contextuels ou bien celle des langages réguliers forment une algèbre de Boole, c'est-à-dire une classe de langages stable par intersection et par complémentation. En revanche, celle des langages algébriques (ou encore celle des langages récursivement énumérables) n'est pas stable par complémentation et de nombreux travaux ont permis de mettre en évidence des sous-classes de langages algébriques stables par complémentation [1, 35, 6, 12], comme par exemple l'algèbre de Boole des langages d'automates à pile visibles [1].

Par ailleurs, il est possible d'associer à chacune des classes de langages définies dans la hiérarchie de Chomsky, une classe de machines abstraites reconnaissant les langages de cette classe. Par exemple, les langages algébriques sont les langages reconnus par les automates à pile, ou encore les langages contextuels sont ceux reconnus par les machines de Turing linéairement bornées [34, 23]. Etant donnée une machine abstraite, on peut considérer son graphe des configurations, dont les arcs sont donnés par les transitions de la machine entre deux configurations. Un mot est alors reconnu par la machine s'il étiquette un chemin acceptant dans son graphe des configurations, c'est-à-dire un chemin dont la source est une configuration initiale et dont le but est une configuration finale.

Cela suggère de considérer la notion d'automate infini, c'est-à-dire de graphe infini muni d'ensembles de sommets initiaux et finaux, et de s'intéresser aux langages

des mots acceptés par cet automate, c'est-à-dire précisément les langages de mots qui étiquettent un chemin dont la source est un sommet initial et dont le but est un sommet final.

Graphe infini de présentation finie

Dans ce manuscrit, les graphes que l'on considère sont orientés, arc-étiquetés et simples au sens où il n'y a pas plusieurs arcs de même source, même but et même étiquette. Les graphes que l'on considère sont potentiellement infinis, c'est-à-dire que l'ensemble des sommets et des arcs sont potentiellement dénombrables. Cependant, ils restent de présentation finie. De manière informelle, cela signifie qu'ils peuvent être décrits par une quantité finie d'information (par exemple un système de réécriture). En particulier, l'ensemble des étiquettes apparaissant dans le graphe est fini.

On peut définir, pour chaque niveau de la hiérarchie de Chomsky, une classe d'automates potentiellement infinis acceptant précisément les langages appartenant au niveau considéré. C'est ainsi que les langages réguliers sont acceptés par les automates finis, les langages algébriques sont acceptés par les automates suffixes de mots (le graphe des configurations d'un automate à pile est un exemple d'automate suffixe de mots et un automate suffixe de mots est essentiellement le graphe des configurations d'un automate à pile) [9], les langages contextuels sont les langages acceptés par les automates rationnels [41] et les langages récursivement énumérables sont les langages acceptés par les graphes des transitions des machines de Turing étiquetées [11].

Les classes d'automates considérés ci-dessus sont définies, a priori, de manière concrète, dans le sens où l'on dispose d'un nommage explicite des sommets du graphe sous-jacent. Par exemple, un automate suffixe de mots est le graphe de réécriture d'un système de réécriture de mots, muni d'ensembles de sommets initiaux et finaux qui sont des langages réguliers de mots. Une autre façon de définir des classes intéressantes de graphes infinis est de procéder par transformations de graphes. Diverses transformations ont été étudiées comme par exemple le dépliage, la structure arborescente ou encore les substitutions inverses. Notons que pour chacune d'elles, les transformés de graphes isomorphes restent isomorphes. Il se trouve que les automates suffixes de mots sont précisément les graphes obtenus par application inverse de substitutions rationnelles aux dépliés des graphes finis [10]. Les transformations logiques sont d'autres transformations intéressantes.

Logique

Chacune des logiques considérées dans ce travail, est la donnée d'une relation de satisfaction entre un ensemble de formules et une classe de structures. Dans ce manuscrit, on s'intéressera à la logique du premier ordre (FO), à la logique du second

ordre monadique (MSO), ainsi qu'à des logiques intermédiaires en termes d'expressivité, et notamment à la logique du premier ordre avec accessibilité (FO[Reach]).

Il est bien connu que la logique FO, où l'on ne s'autorise à quantifier que sur des sommets de la structure, ne permet d'exprimer que des propriétés locales. La logique MSO, en permettant de quantifier sur des parties de la structure, est bien plus expressive. Par exemple, la connexité dans les graphes est exprimée par une formule MSO. De nombreux travaux ont pour objectif de déterminer des structures (ici des automates) dont la théorie relativement à une logique L donnée est décidable. Autrement dit, on essaie de déterminer des structures pour lesquelles il existe une procédure effective qui calcule les énoncés de L satisfaits par la structure.

C'est ainsi que Rabin a montré dans [40] que la théorie du second ordre monadique de l'arbre binaire infini est décidable. Notons que, précédemment, Büchi avait montré dans [3] que la théorie du second ordre monadique de la demi-droite définie par la relation successeur, est décidable. Dès lors que la théorie MSO d'un graphe G est décidable, la théorie MSO de tout graphe qui en est MSO-interprétation, c'est-à-dire dont les arcs sont obtenus à partir de ceux de G comme satisfaisant certaines MSO-formules, reste décidable. La théorie MSO de tout automate suffixe de mots est ainsi décidable. En effet, Muller et Schupp ont montré dans [33] que le graphe des configurations d'un automate à pile est MSO-interprétation de l'arbre binaire infini. Plus généralement, et comme Courcelle a montré dans [15] que le dépliage depuis un sommet MSO-définissable préserve la décidabilité de la logique MSO, Caucal a défini une hiérarchie de graphes, la hiérarchie à pile, dont l'une des propriétés remarquable est que la MSO-théorie de tout graphe lui appartenant est décidable [8]. Le premier niveau de cette hiérarchie est constitué des MSO-interprétations des dépliés des graphes finis et le niveau $n > 1$ est constitué des MSO-interprétations des dépliés du niveau précédent. D'autres caractérisations de cette hiérarchie existent, que ce soit en terme de transformations de graphes ou en terme de présentations concrètes [5]. De plus, diverses extensions de cette hiérarchie ont été obtenues [14, 37]. En général, ces extensions ne préservent pas la décidabilité de la logique MSO. L'une de ces extensions, la hiérarchie arbre-automatique, consiste en une hiérarchie de graphes dont la FO-théorie reste cependant décidable : chaque niveau est obtenu à partir d'un générateur du niveau correspondant de la hiérarchie à pile par interprétation à ensembles finis. Une interprétation à ensembles finis d'un graphe G n'est qu'une FO-interprétation du graphe des parties finies de G [14, 37].

Point de départ de notre travail : le quart de la grille infinie

Le quart de la grille infinie n'appartient pas à la hiérarchie à pile car sa théorie MSO n'est pas décidable. En effet, le problème indécidable de l'arrêt des machines de Turing s'y réduit. Cependant, la FO[Reach]-théorie du quart de la grille infinie est décidable [17]. Ce résultat provient du fait que :

— le quart de la grille infinie est concrètement présentable comme le graphe de réécriture suffixe d'un système de réécriture de termes clos (graphe GTR)

— tout graphe GTR, enrichi de la relation d'accessibilité, appartient au premier niveau de la hiérarchie arbre-automatique ; en particulier, la FO[Reach]-théorie d'un graphe GTR est décidable.

Ce résultat constitue le point de départ de notre travail. En effet, **se pose la question de pouvoir mettre en évidence, par transformations de graphes, des sous-classes de graphes du premier niveau arbre-automatique dont la** FO[Reach]**-théorie reste décidable**. Ce questionnement est aussi motivé par des résultats obtenus par Morvan dans [4] que nous allons rappeler.

Morvan a notamment considéré les graphes rationnels (il existe un graphe rationnel dont la FO-théorie n'est pas décidable), dont une sous-classe est constituée des graphes mot-automatiques ; un graphe est mot-automatique si chacune de ses relations est acceptée par un transducteur synchronisé fini. Un graphe mot-automatique appartient au premier niveau arbre-automatique et sa FO-théorie est donc décidable [21]. Cependant, la FO[Reach]-théorie d'un graphe mot-automatique peut ne pas être décidable. Il se trouve que le quart de la grille infinie est un DAG (directed acyclic graph) mot-automatique mais qu'il existe des DAG rationnels dont la FO-théorie n'est pas décidable. **Il se pose donc la question de pouvoir mettre en évidence, par transformations de graphes, une classe de DAG mot-automatiques dont la** FO[Reach]**-théorie est décidable**.

Traces de Mazurkiewicz

Le quart de la grille infinie peut être présenté concrètement comme le graphe des configurations d'un système à deux compteurs, que l'on peut incrémenter de manière indépendante. Une notion permettant de décrire plus généralement le comportement non-séquentiel de systèmes concurrents est celle de traces de Mazurkiewicz [28]. Dans ce travail, on introduit notamment la notion de *régularité par niveau de langages de traces*. Un langage de traces est régulier par niveaux si l'ensemble des formes normales de Foata de ses éléments constitue un langage régulier de mots. Nous observerons que la régularité par niveaux est une notion de reconnaissabilité (dans le monoïde des traces) plus faible que celle définie dans un monoïde quelconque par Eilenberg [19].

Plan et contributions

Les contributions feront l'objet des chapitres 5 et 6.

Dans le chapitre 2, on rappelle diverses notions préliminaires utiles pour la lecture du manuscrit.

Dans le chapitre 3, on rappelle divers résultats concernant la hiérarchie de langages définie par Chomsky, les automates infinis et les transformations de graphes.

Dans le chapitre 4, après avoir rappelé diverses notions de base concernant les traces de Mazurkiewicz, on introduit la notion de régularité par niveau de langage de traces.

Dans le chapitre 5, on présente une première contribution [27]. Nous considérons des systèmes reconnaissables de réécriture de traces, à contextes réguliers par niveaux (RTL). Nous prouvons que tout automate dont le graphe sous-jacent est le graphe de réécriture d'un système RTL et dont les ensembles de sommets initiaux et finaux sont réguliers par niveaux (automate RTL), est mot-automatique. En particulier, la FO-théorie d'un automate RTL est décidable. Nous prouvons également que le problème de la vacuité du langage accepté par un automate RTL est indécidable. Ensuite, nous prouvons que la FO[Reach]-théorie de tout automate dont le graphe sous-jacent est le déplié concurrent d'un graphe fini concurrent et dont les ensembles de sommets initiaux et finaux sont réguliers par niveaux, est décidable. Le quart de la grille infinie est le déplié concurrent d'un graphe fini concurrent. Nous mettons également en évidence divers graphes finis concurrents dont les dépliés concurrents ne sont pas des graphes GTR (contrairement au quart de la grille infinie), dont nous avons déjà rappelé que la FO[Reach]-théorie était décidable. Un exemple de tel déplié est l'arbre du quart de la grille infinie. Ainsi, la classe des dépliés concurrents des graphes finis concurrents constitue une classe de DAG mot-automatiques dont la FO[Reach]-théorie est décidable.

Dans le chapitre 6, on présente une deuxième contribution [26]. Nous considérons des automates de traces. Leurs sommets sont des traces de Mazurkiewicz et les langages acceptés sont bien des langages de mots. Ici, la longueur d'une trace étant donnée par la longueur de sa forme normale de Foata, nous dirons qu'un automate de traces G est *longueur-réductible* en un automate de traces H, s'il existe un morphisme de G dans H préservant la longueur. Nous définissons pour les automates de traces deux opérations que sont la synchronisation par niveaux et la superposition par niveaux et nous montrons que si une famille $\mathcal{F}$ d'automates de traces est fermée pour ces opérations, alors pour tout automate déterministe $H \in \mathcal{F}$, les langages acceptés par les automates déterministes de $\mathcal{F}$ qui sont longueur-réductibles en H forment une algèbre de Boole. Nous montrons ensuite que la classe TrSuffix des automates suffixes de traces à contextes réguliers par niveaux satisfait ces propriétés de fermeture. La classe TrSuffix est l'extension aux traces des automates suffixes de mots. Dans ce chapitre, nous appelons réseau de Petri généralisé un automate suffixe de traces sur un alphabet de dépendance pour lequel la dépendance est réduite à l'égalité. Il s'agit d'une notion plus générale que celle des réseaux de Petri usuels, dont nous discuterons. Nous montrons alors que la sous-famille $\mathrm{TrSuffix}^{\mathrm{Petri}} \subseteq \mathrm{TrSuffix}$ des réseaux de Petri généralisés satisfait également les propriétés de fermeture énoncées ci-dessus. Cela conduit notamment à diverses algèbres de Boole de langages acceptés par des réseaux de Petri généralisés déterministes.

Chapitre 2

Préliminaires

2.1 Préliminaires ensemblistes

2.1.1 Ensembles

L'ensemble vide est noté $\varnothing$. L'ensemble des entiers naturels est noté $\mathbb{N}$. L'ensemble des entiers relatifs est noté $\mathbb{Z}$.

Etant donné un ensemble E, l'ensemble des parties (respectivement l'ensemble des parties finies) de E est noté $\mathcal{P}(E)$ (respectivement $\wp(E)$).

Les opérations d'union, d'intersection, de complémentation et de produit d'ensembles sont définies de manière usuelle. Etant donné un ensemble E, le complémentaire d'une partie P de E (dans E) sera notée $E \setminus P$.

Le cardinal de E sera noté $\mathrm{card}(E)$.

Un vecteur d'entiers naturels (respectivement d'entiers relatifs) de dimension $p \in \mathbb{N}$ est un élément de $\mathbb{N}^p$ (respectivement $\mathbb{Z}^p$).

2.1.2 Relations n-aire

Soit $n > 1$. Une *relation n-aire* $\mathcal{R}$ est une partie de $E_1 \times \cdots \times E_n$, où $E_1, \ldots, E_n$ sont des ensembles. Si $E_1 = \cdots = E_n$, on dit que $\mathcal{R}$ est une relation n-aire sur E.

Si $\mathcal{R}$ est une relation binaire (2-aire), alors le *domaine* et l'*image* de $\mathcal{R}$ sont définis respectivement par :

$$\begin{array}{rcl} \mathrm{Dom}(\mathcal{R}) & := & \{e_1 \in E_1 \mid \exists\ e_2 \in E_2\ (e_1, e_2) \in \mathcal{R}\} \\ \mathrm{Im}(\mathcal{R}) & := & \{e_2 \in E_2 \mid \exists\ e_1 \in E_1\ (e_1, e_2) \in \mathcal{R}\} \end{array}$$

Si $\mathcal{R}$ est une relation binaire sur un ensemble E, rappelons que :

— $\mathcal{R}$ est *réflexive* si

$$\forall e \in E\ (e, e) \in \mathcal{R}$$

— $\mathcal{R}$ est *symétrique* si

$$\forall e \in E\ \forall f \in E\ ((e, f) \in \mathcal{R} \longrightarrow (f, e) \in \mathcal{R})$$

— $\mathcal{R}$ est *anti-symétrique* si

$$\forall e \in E \; \forall f \in E \; (((e,f) \in \mathcal{R} \; \wedge \; (f,e) \in \mathcal{R}) \; \longrightarrow \; e = f)$$

— $\mathcal{R}$ est *transitive* si

$$\forall e \in E \; \forall f \in E \; \forall g \in E \; (((e,f) \in \mathcal{R} \; \wedge \; (f,g) \in \mathcal{R}) \; \longrightarrow \; (e,g) \in \mathcal{R})$$

Une relation d'*équivalence* $\mathcal{R}$ sur E est une relation binaire réflexive, symétrique et transitive. La *classe d'équivalence* de $x \in E$ pour $\mathcal{R}$ est

$$[x] := \{y \in E \mid (x,y) \in \mathcal{R}\}$$

Le *quotient* $E/\mathcal{R}$ de E par $\mathcal{R}$ est alors l'ensemble des classes d'équivalences :

$$E/\mathcal{R} := \{[x] \mid x \in E\}$$

Une relation d'*ordre* (partiel) sur E est une relation binaire réflexive, anti-symétrique et transitive.

Dans la suite, pour une relation binaire $\mathcal{R}$, on notera $e\mathcal{R}f$ au lieu de $(e,f) \in \mathcal{R}$.

2.1.3 Fonctions

Les fonctions considérées ici sont à priori partielles.

Une *fonction* f est une relation binaire telle que

$$((x,y) \in f \; \wedge \; (x,y') \in f) \; \longrightarrow \; y = y'$$

Si $(x,y) \in f$, alors y est l'unique *image* de x par f et est notée $f(x)$.

Si la fonction f est une partie de $E_1 \times E_2$, alors on dira que f est une fonction de E_1 dans E_2 et on écrira $f : E_1 \longrightarrow E_2$.

Etant donné un ensemble P, la *restriction de* f *à* P est la fonction

$$f_{|P} := \{(x,y) \in f \mid x \in P\}$$

Les notions de fonction injective, surjective et bijective sont définies de manière usuelle :

— f est *injective* si

$$\forall x, x' \in E_1 \; (f(x) = f(x') \; \longrightarrow \; x = x')$$

— f est *surjective* si

$$\forall y \in E_2 \; \exists x \in E_1 \; f(x) = y$$

— f est *bijective* si f est injective et surjective.

Une *application* f de E_1 dans E_2 est une fonction de E_1 dans E_2 telle que

$$\mathrm{Dom}(f) = E_1$$

2.2 Monoïdes

2.2.1 Généralités

Soit E un ensemble. Une *opération interne* op dans E est une application de $E \times E$ dans E. L'image par op de $(x, y) \in E \times E$ est notée x op y. L'opération op est *associative* si $(x \text{ op } y) \text{ op } z = x \text{ op } (y \text{ op } z)$ pour tous $x, y, z \in E$.

Un *élément neutre* pour op est un élément $e \in E$ tel que pour tout $x \in E$, $e \text{ op } x = x \text{ op } e = x$. Une opération a au plus un élément neutre.

Un *monoïde* est un couple (M, op), où M est un ensemble et op est une opération interne dans M vérifiant :

— op est associative,

— op possède un élément neutre noté 1.

L'opération interne op est appelée *produit* du monoïde (M, op). S'il n'y a pas d'ambiguïté, le produit de deux éléments $x, y \in M$ est noté plus simplement xy.

Dans la suite, on s'autorisera l'abus qui consiste, étant donné un ensemble M, de parler du « monoïde M » si l'opération interne sur M est connue ou s'il n'y a pas d'ambiguïté sur cette opération.

Etant donnés deux monoïdes M_1 et M_2, une fonction f de M_1 dans M_2 est un *morphisme de monoïdes* si

— $f(1_{M_1}) = 1_{M_2}$

— pour tous $m, m' \in M_1$, $f(mm') = f(m)f(m')$.

Si de plus f est bijective, on dit que c'est un isomorphisme de monoïdes.

Sous-monoïde, monoïde engendré par une partie

Soit N une partie de M telle que

$$1 \in N \text{ et } \forall x \in N \ \forall y \in N \ xy \in N$$

Alors le couple $(N, \text{op}_{|N \times N})$ est un monoïde. Un tel monoïde est un *sous-monoïde* de M.

Soit P une partie de M. L'ensemble des sous-monoïdes de M contenant P est stable par intersection. Le *monoïde engendré* par P dans M est le plus petit sous-monoïde de M contenant P. Un monoïde M est *finiment engendré* s'il existe une partie finie P telle que le monoïde engendré par P dans M est précisément M.

Exemple 2.2.1. Soit E un ensemble. L'ensemble des parties de E muni de l'opération d'union est un monoïde d'élément neutre la partie vide. L'ensemble des parties finies de E est un monoïde finiment engendré par l'ensemble des singletons de E.

2.2.2 Monoïde libre

Un *alphabet* est un ensemble fini de *lettres*.

Soit Σ un alphabet. Un *mot* u sur Σ est une suite finie d'éléments de Σ, notée $u = u_1 \dots u_{|u|}$, où $|u|$ est la longueur ou taille de u. Le *mot vide* est la suite de taille 0, noté ε. On note Σ^* l'ensemble des mots sur Σ et $\Sigma^+ := \Sigma^* \setminus \{\varepsilon\}$ l'ensemble des mots sur Σ distincts du mot vide. Pour tout $a \in \Sigma$ et tout $u \in \Sigma^*$, on note $|u|_a$ le nombre d'occurences de la lettre a dans u.

Etant donné deux mots $u = u_1 \dots u_m$ et $v = v_1 \dots v_n$, leur *concaténation*, notée uv, est le mot sur Σ défini par :

$$uv := u_1 \dots u_m v_1 \dots v_n$$

Muni de la concaténation, Σ^* est un monoïde d'élément neutre ε et engendré par Σ, appelé monoïde libre sur Σ.

Un *langage de mots sur* Σ est une partie de Σ^*.

2.2.3 Monoïde des relations binaires sur un ensemble

Etant données des relations binaires $\mathcal{R}_1$ et $\mathcal{R}_2$ sur E, leur *composée*, notée $\mathcal{R}_1\mathcal{R}_2$, est la relation binaire sur E définie par :

$$\mathcal{R}_1\mathcal{R}_2 := \{(e,g) \in E^2 \mid \exists\ f \in E\ ((e,f) \in \mathcal{R}_1\ \wedge\ (f,g) \in \mathcal{R}_2)\}$$

La relation $\mathrm{Id}_E := \{(e,e) \in E^2 \mid e \in E\}$ est élément neutre pour l'opération de composition. Muni de la composition, l'ensemble des relations binaires sur E est un monoïde.

2.2.4 Monoïde produit

Soit (M_1, op_{M_1}) et (M_2, op_{M_2}) deux monoïdes.

Le *monoïde produit* de M_1 par M_2 est le monoïde $(M_1 \times M_2, \mathrm{op}_{M_1 \times M_2})$, où l'opération interne $\mathrm{op}_{M_1 \times M_2}$ est définie pour tous $(x_1, x_2), (y_1, y_2) \in M_1 \times M_2$ par

$$(x_1, x_2)\ \mathrm{op}_{M_1 \times M_2}\ (y_1, y_2) := (x_1\ \mathrm{op}_{M_1}\ y_1, x_2\ \mathrm{op}_{M_2}\ y_2)$$

2.2.5 Monoïde quotient

Soit (M, op_M) un monoïde. Une *congruence* $\equiv$ sur M est une relation d'équivalence sur M telle que

$$(x_1 \equiv x_2\ \wedge\ y_1 \equiv y_2)\ \longrightarrow\ x_1 y_1 \equiv x_2 y_2$$

Le *monoïde quotient* de M par $\equiv$ est le monoïde $(M/\equiv, \mathrm{op}_{M/\equiv})$, où l'opération interne $\mathrm{op}_{M/\equiv}$ est définie pour tout $[x], [y] \in M/\equiv$ par

$$[x]\ \mathrm{op}_{M/\equiv}\ [y] := [xy]$$

2.2.6 Parties rationnelles, parties reconnaissables

Un *langage* du monoïde M est une partie de M.

Le produit de concaténation de deux langages L_1 et L_2 de M est le langage L_1L_2 défini par

$$L_1L_2 := \{x_1x_2 \mid x_1 \in L_1, x_2 \in L_2\}$$

En particulier, $L\varnothing = \varnothing L = \varnothing$ pour tout langage L de M.

Etant donné $x \in M$ et un langage L de M, le produit de L par le langage singleton $\{x\}$ (respectivement le produit du langage singleton $\{x\}$ par L) est noté plus simplement Lx (respectivement xL).

Observons que $\mathcal{P}(M)$, muni du produit de concaténation de langages, est un monoïde d'élément neutre le langage singleton $\{1\}$.

Soit L un langage de M. Les puissances de L sont définies en posant : $L^0 := \{1\}$ et pour $n \geqslant 1$, $L^n := L^{n-1}L$.

L'*étoile* de L est le langage L^* de M défini par :

$$L^* := \bigcup_{n\geqslant 0} L^n$$

Ainsi L^* est le plus petit langage contenant $L \cup \{1\}$ et fermé par produit. Il s'agit du sous-monoïde de M engendré par L.

Un langage L de M est *rationnel* s'il appartient au plus petit ensemble $\mathrm{Rat}(M)$ des parties de M qui satisfait les conditions suivantes :

— $\mathrm{Rat}(M)$ contient la partie vide $\varnothing$ ainsi que les singletons de $\mathcal{P}(M)$;

— $\mathrm{Rat}(M)$ est fermé par union, produit et étoile. Autrement dit, si $L_1, L_2 \in \mathrm{Rat}(M)$, alors $L_1 \bigcup L_2, L_1L_2, L_1^* \in \mathrm{Rat}(M)$.

Un langage rationnel de M peut être décrit à l'aide d'une *expression rationnelle* sur M. L'ensemble ExpRat_Σ des *expressions rationnelles* sur M est défini par induction de la manière suivante :

$$\mathrm{ExpRat}_\Sigma := \quad \varnothing \mid x \mid (e_1 + e_2) \mid (e_1 \cdot e_2) \mid (e^*) \mid \quad (x \text{ décrit } M)$$

Rappelons à présent la notion de reconnaissabilité au sens d'Eilenberg pour les parties d'un monoïde [19]. Un langage L de M est *reconnaissable* s'il existe un monoïde fini N et un morphisme de monoïdes $\phi : M \longrightarrow N$ tel que $L = \phi^{-1}[\phi[L]]$.

Théorème 2.2.2 ([29])**.** *Les assertions suivantes sont équivalentes :*

1. *M est finiment engendré,*
2. *tout langage reconnaissable de M est rationnel,*
3. *l'ensemble M est un langage rationnel de M.*

Langages réguliers de mots

Dans le cas du monoïde libre Σ^*, les parties rationnelles et les parties reconnaissables coïncident.

Théorème 2.2.3 (Kleene)**.** *Soit Σ un alphabet. Un langage de Σ-mots $L \subseteq \Sigma^*$ est rationnel si et seulement s'il est reconnaissable.*

Un langage rationnel (ou reconnaissable) de mots est dit *régulier*. Notons $\mathrm{Reg}(\Sigma^*)$ la classe des langages réguliers de Σ-mots.

Résiduation et reconnaissabilité

Etant donné un langage $L \subseteq M$ et $t \in M$, le *résidu à droite* (respectivement le *résidu à gauche*) de L par t, noté Lt^{-1} (respectivement $t^{-1}L$), est

$$Lt^{-1} := \{s \in M \mid st \in L\} \text{ (respectivement } t^{-1}L := \{s \in M \mid ts \in L\})$$

Proposition 2.2.4. *Les assertions suivantes sont équivalentes*

1. *L est reconnaissable,*
2. *l'ensemble $\{t^{-1}L \mid t \in M\}$ des résidus à gauche de L est fini,*
3. *l'ensemble $\{Lt^{-1} \mid t \in M\}$ des résidus à droite de L est fini.*

Considérons un alphabet Σ et $\pi : \Sigma^* \to M$ un morphisme surjectif. La proposition suivante montre que la reconnaissabilité d'un langage d'un monoïde est équivalente à la régularité de l'ensemble des Σ-codages de ses éléments.

Proposition 2.2.5. *L est reconnaissable si et seulement si $\pi^{-1}[L]$ est régulier.*

2.3 Structures relationnelles, logiques

2.3.1 Définition d'une structure relationnelle

Une *signature relationnelle* σ est un ensemble de symboles de relation R, chacun muni d'une arité $a(R)$, un entier naturel. Une *σ-structure relationnelle* $\mathfrak{A}$ est la donnée d'un ensemble A, son domaine, et pour chaque $R \in \sigma$, d'une relation $a(R)$-aire sur A (l'interprétation de R dans $\mathfrak{A}$), c'est-à-dire d'une partie $R^{\mathfrak{A}} \subseteq A^{a(R)}$.

Exemple 2.3.1. Le corps des nombres réels $\mathbb{R}$ est une structure relationnelle sur la signature relationnelle $\sigma = \{0, 1, +, \times\}$, où 0 et 1 sont des *constantes*, c'est-à-dire des symboles d'arité nulle, $+$ et $\times$ sont des symboles d'arité 3. L'interprétation du symbole 0 (respectivement du symbole 1) est le nombre réel 0 (respectivement le nombre réel 1). L'interprétation du symbole $+$ (respectivement du symbole $\times$) est l'ensemble des triplets de nombre réels (x, y, z) tels que $z = x + y$ (respectivement $z = x \times y$).

2.3.2 Logique du premier ordre (FO)

On commence par définir les formules de la logique du premier ordre relativement à σ.

On suppose que l'on dispose d'un ensemble $\mathcal{V}_1 = \{x, x_1, x_2, y, z, \ldots\}$ de *variables du premier ordre*, qui seront interprétées par des éléments du domaine et de symboles logiques que sont les connecteurs $\neg$, $\wedge$ et le quantificateur $\exists$.

Les formules de la logique du premier ordre (FO-formules) sont définies récursivement à partir des FO-formules atomiques.

— Les FO-formules atomiques sont : $R(x_1, \ldots, x_{a(R)})$ où $R \in \sigma$.

— Si ϕ et ψ sont des FO-formules, alors $(\phi \wedge \psi)$, $\neg\phi$ et $\exists x\phi$ sont des FO-formules.

Les *variables libres* de $R(x_1, \ldots, x_{a(R)})$ sont $x_1, \ldots, x_{a(R)}$. Les variables libres de $\phi \wedge \psi$ (respectivement $\neg\phi$) sont celles de ϕ et de ψ (respectivement ϕ). Les variables libres de $\exists x\phi$ sont celles de ϕ, à l'exception de la variable x. On note $\phi(x_1, \ldots, x_n)$ une FO-formule dont les variable libres figurent parmi les variables $x_1, \ldots, x_n$. Un FO-*énoncé* est une FO-formule sans variable libre.

On définit à présent la sémantique des FO-formules. Plus précisément, on définit une relation de satisfaction entre les FO-formules $\phi(x_1, \ldots, x_n)$ et les couples $(\mathfrak{A}, \nu)$, où $\mathfrak{A}$ est une σ-structure de domaine A et $\nu : \mathcal{V}_1 \longrightarrow A$ une *valuation*. Cette relation de satisfaction sera notée $\models$.

Soit $\mathfrak{A}$ une σ-structure et ν une valuation.

— $(\mathfrak{A}, \nu) \models R(x_1, \ldots, x_{a(R)})$ si $(\nu(x_1), \ldots, \nu(x_{a(R)})) \in R^{\mathfrak{A}}$

— $(\mathfrak{A}, \nu) \models (\phi \wedge \psi)$ si $(\mathfrak{A}, \nu) \models \phi$ et $(\mathfrak{A}, \nu) \models \psi$

— $(\mathfrak{A}, \nu) \models \neg\phi$ si $(\mathfrak{A}, \nu) \not\models \phi$

— $(\mathfrak{A}, \nu) \models \exists x\phi$ s'il existe un élément $a \in A$ de sorte que $(\mathfrak{A}, \nu') \models \phi$, où $\nu'(x) = a$ et $\nu'(y) = \nu(y)$ pour tout $y \neq x$.

La *théorie du premier ordre* (FO-théorie) de $\mathfrak{A}$ est l'ensemble des FO-énoncés satisfaits par $\mathfrak{A}$.

Dans la suite, au lieu d'écrire $(\mathfrak{A}, \nu) \models \phi(x_1, \ldots, x_n)$, où $\nu(x_i) = a_i$ $(1 \leqslant i \leqslant n)$, on s'autorisera l'abus d'écriture $\mathfrak{A} \models \phi(a_1, \ldots, a_n)$.

Exemple 2.3.2. Reprenons l'exemple du corps des nombres réels $\mathbb{R}$, vu comme une $\{0, 1, +, \times\}$-structure relationnelle. On dispose de :

$$(\mathbb{R}, \nu) \models \exists y\ x \times y = 1 \text{ où } \nu(x) \text{ est n'importe quel nombre réel non nul}$$

où $x \times y = 1$ est un raccourci d'écriture pour $\times(x, y, 1)$. En revanche, si $\nu(x) = 0$, alors

$$(\mathbb{R}, \nu) \not\models \exists y\ x \times y = 1$$

et même

$$(\mathbb{R}, \nu) \models \neg\ \exists y\ x \times y = 1$$

2.3.3 Logique du second ordre monadique (MSO)

On enrichit le langage du premier ordre en supposant que l'on dispose d'un ensemble $\mathcal{V}_2 = \{X, X_1, X_2, Y, Z, \ldots\}$ de *variables du second ordre monadique* (qui seront interprétées par des parties du domaine) et du symbole $\in$.

Les formules de la logique du second ordre monadique (MSO) sont obtenues d'abord en ajoutant aux formules atomiques du premier ordre les formules de la forme $x \in X$, ensuite en ajoutant aux formules du premier ordre le cas suivant $\exists X\phi$, où ϕ est une formule de la logique du second ordre monadique.

Une valuation est dorénavant une fonction ν définie sur $\mathcal{V}_1 \dot{\cup} \mathcal{V}_2$, qui à une variable du premier ordre associe un élément du domaine A et à une variable du second ordre monadique associe une partie de A.

On étend de manière usuelle la notion de variable libre aux variables monadiques du second ordre.

La relation de satisfaction est alors étendue de la manière suivante :

— $(\mathfrak{A}, \nu) \models x \in X$ si $\nu(x) \in \nu(X)$

— $(\mathfrak{A}, \nu) \models \exists X\phi$ s'il existe une partie $P \subseteq A$ telle que $(\mathfrak{A}, \nu') \models \phi$, où $\nu'(X) = P$ et les restrictions de ν' et ν coincident sur $\mathcal{V}_1 \dot{\cup} \mathcal{V}_2 \setminus \{X\}$.

La théorie du second ordre monadique (MSO-théorie) de $\mathfrak{A}$ est l'ensemble des MSO-énoncés satisfaits par $\mathfrak{A}$.

Si l'on restreint les interprétations des variables monadiques du second ordre aux parties finies du domaine, alors on définit la *logique du second ordre monadique faible* (WMSO).

Exemple 2.3.3. L'ensemble des entiers naturels $\mathbb{N}$ est une structure relationnelle sur $\{0, S\}$, où le symbole 0 est interprété par l'entier naturel 0 et le symbole binaire S par l'ensemble des couples d'entiers naturels consécutifs $S^{\mathbb{N}} = \{(n, n+1) \mid n \in N\}$. Considérons la formule du second ordre monadique ϕ définie par

$$\phi(Z) = \neg\, \exists z\ (z \in Z\ \wedge\ \exists z'\ (S(z, z')\ \wedge\ \neg\, z' \in Z))$$

On a

$$(\mathbb{N}, \nu) \models \phi(Z)$$

où $\nu(Z)$ est l'ensemble vide ou bien n'importe quelle section finale de $\mathbb{N}$, c'est-à-dire une partie de $\mathbb{N}$ de la forme $P_a = \{n \in \mathbb{N} \mid n \geqslant a\}$, avec $a \in \mathbb{N}$.

Dans la suite, on s'autorisera, comme à l'accoutumée, à faire usage du quantificateur supplémentaire $\forall$ ainsi que des connecteur supplémentaire $\longrightarrow$ et $\vee$. En particulier, on utilisera les raccourcis d'écriture suivants :

— $\forall x\ \phi$ pour $\neg\, \exists x\, \neg\, \phi$

— $(\phi\ \vee\ \psi)$ pour $\neg\, (\neg\, \phi\ \wedge\ \neg\, \psi)$

— $(\phi\ \longrightarrow\ \psi)$ pour $\neg\, (\phi\ \wedge\ \neg\, \psi)$.

2.4 Graphes

2.4.1 Définition, généralités

Les graphes que l'on considère sont orientés, arc-étiquetés et simples au sens où il n'y a pas plusieurs arcs de même source, même but et même étiquette.

Soit Λ un alphabet. Un Λ-*graphe* G sur un ensemble V (de sommets), est une partie de $V \times \Lambda \times V$ d'ensemble de sommets

$$V_G := \{v \in V \mid \exists \lambda, w \ (w, \lambda, v) \in G \text{ ou } (v, \lambda, w) \in G\}$$

et d'ensemble d'*étiquettes*

$$\Lambda_G = \{\lambda \in \Lambda \mid \exists v, w \in V \ (v, \lambda, w) \in G\}$$

Le graphe G est *fini* si son ensemble de sommets V_G est fini.

Le *degré sortant* (respectivement le *degré entrant*) d'un sommet $v_0 \in V_G$ est $d^+(v_0) := \text{card}(\{w \mid \exists \lambda, w \ (v_0, \lambda, w) \in G\})$ (respectivement $d^-(v_0) := \text{card}(\{w \mid \exists \lambda, w \ (w, \lambda, v_0) \in G\})$). Le *degré* d'un sommet $v_0 \in V_G$ est

$$d(v_0) := d^+(v_0) + d^-(v_0)$$

Le graphe G est de *degré fini* si le degré de chacun de ses sommets est fini et de *degré borné* s'il existe un entier naturel M tel que chacun de ses sommets soit de degré inférieur ou égal à M.

Un graphe G est une structure relationnelle, encore notée abusivement G, relativement à la signature relationnelle $\{E_\lambda \mid \lambda \in \Lambda\} \cup \{=\}$: son domaine est V_G, l'interprétation du symbole binaire $=$ est naturellement la relation binaire d'égalité sur les sommets de G

$$=^G = \{(v, v) \mid v \in V_G\}$$

et pour chaque $\lambda \in \Lambda$, l'interprétation de E_λ dans G est la relation binaire E_λ^G définie par

$$E_\lambda^G = \{(v, w) \mid \exists \lambda \ (v, \lambda, w) \in G\}$$

Un élément $(v, \lambda, w) \in G$ est un *arc*. Son *étiquette* est λ, sa *source* est v et son *but* est w. La notation $v \xrightarrow[G]{\lambda} w$ (ou $v \xrightarrow{\lambda} w$ lorsque G est sous-entendu) signifie $(v, \lambda, w) \in G$. La notation $v \underset{G}{\text{---}} w$ signifie qu'il existe $\lambda \in \Lambda_G$ tel que $v \xrightarrow[G]{\lambda} w$ ou bien $w \xrightarrow[G]{\lambda} v$.

La *restriction de* G *à* $P \subseteq V_G$ est

$$G_{|P} := \{(v, \lambda, w) \mid (v, \lambda, w) \in G \ \wedge \ v, w \in P\}$$

Le graphe G est *déterministe* si pour tout $\lambda \in \Lambda$

$$(v \xrightarrow[G]{\lambda} w \ \wedge \ v \xrightarrow[G]{\lambda} w') \ \longrightarrow \ w = w'$$

.

Exemple 2.4.1. Un graphe déterministe G satisfait le FO-énoncé suivant.

$$\bigwedge_{\lambda \in \Lambda_G} \forall x \ (\exists y \ \exists y' \ (x \xrightarrow{\lambda} y \wedge x \xrightarrow{\lambda} y') \ \longrightarrow \ y = y')$$

Un *chemin* dans G de *source* v et de *but* w, étiqueté par un mot $\lambda_1 \dots \lambda_k \in \Lambda^*$, est une suite finie de la forme $v \xrightarrow[G]{\lambda_1} v_1 \dots v_{k-1} \xrightarrow[G]{\lambda_k} w$, avec $v = w$ pour $k = 0$. Nous notons $v \xrightarrow[G]{\lambda_1 \dots \lambda_k} w$ pour signifier l'existence d'un tel chemin. Etant donné un langage de Λ-mots L, nous noterons également $v \xrightarrow[G]{L} w$ pour signifier qu'il existe un mot appartenant à L qui étiquette un chemin dans G de source v et de but w.

Une *boucle* (dans G) en $v \in V_G$ est un chemin de la forme $v \xrightarrow[G]{\lambda} v$.

Si G est fini, alors pour tous $v, w \in V_G$, le langage des Λ-mots qui étiquettent les chemins de source v et de but w

$$L_{v,w} := \{u \in \Lambda^* \mid v \xrightarrow[G]{u} w\}$$

est régulier.

On écrit $v \xrightarrow[G]{*} w$ s'il existe un mot $u \in \Lambda^*$ tel que $v \xrightarrow[G]{u} w$. Notons G_* le $\Lambda \dot\cup \{*\}$-graphe défini par $G_* = G \cup \{(v, *, w) \mid v \xrightarrow[G]{*} w\}$. Le graphe G_* est obtenu à partir de G en ajoutant la relation d'accessibilité.

Un isomorphisme f entre (G, P) et (H, Q), où G et H sont des Λ-graphes et P (respectivement Q) est une partie de V_G (respectivement V_H), est une bijection de V_G sur V_H telle que

$$f[P] = Q \ \text{ et } \ (v \xrightarrow[G]{\lambda} w \iff f(v) \xrightarrow[H]{\lambda} f(w))$$

Graphes connexes, arbres, DAG

Un graphe G est connexe si pour tous sommets distincts $v, w \in V_G$, il existe un chemin de source v et de but w dans le graphe $\{(v, \lambda, w) \mid v \xrightarrow[G]{\lambda} w \text{ ou } w \xrightarrow[G]{\lambda} v\}$.

Un *arbre* est un graphe connexe qui vérifie les propriétés suivantes :

— il existe un sommet, appelé *racine*, qui n'est but d'aucun arc

— tout sommet est but d'au plus un arc.

Un *DAG* est un graphe tel que tout sommet n'est pas source d'un chemin non vide dont le but est lui-même.

Graphes avec ε-arc

On pourra aussi être amené à considérer des graphes dont les arcs sont étiquetés par le mot vide et à en considérer l'ε-fermeture. Plus précisément, soit $\varepsilon \in \Lambda$ une étiquette et G un graphe. L'ε-fermeture de G est le graphe noté $\overline{G}^{\varepsilon}$ défini par

$$\overline{G}^{\varepsilon} := \{v \xrightarrow{\lambda} w \mid \lambda \in \Lambda, v \xrightarrow[G]{\varepsilon^*\lambda\varepsilon^*} w\}$$

2.4.2 Logique du premier ordre avec accessibilité (FO[Reach])

Etant donné un graphe G, le graphe G_* est une structure relationnelle relativement à la signature de G enrichie du nouveau symbole binaire $*$. Ce nouveau symbole est interprété dans G par sa relation d'accessibilité.

La *théorie du premier ordre avec accessibilité* de G (FO[Reach]-théorie de G), est la théorie du premier ordre de G_*.

Exemple 2.4.2. Soit G un graphe. Il existe un sommet de G à partir duquel on peut atteindre un sommet sans successeur si et seulement si G satisfait le FO[Reach]-énoncé suivant

$$\exists x\ \exists y\ (x \xrightarrow{*} y\ \wedge \bigwedge_{\lambda \in \lambda} \neg\ \exists y'\ y \xrightarrow{\lambda} y')$$

Les formules de la *logique du premier ordre avec accessibilité rationnelle* sont obtenues en ajoutant aux formules atomiques du premier ordre les formules de la forme $x \xrightarrow{L} y$, où L est un langage rationnel sur Σ et en étendant la relation de satisfaction de la manière suivante : $(G, \nu) \models x \xrightarrow{L} y$ s'il existe un mot $u \in L$ tel que $\nu(x) \xrightarrow[G]{u} \nu(y)$. La *théorie du premier ordre avec accessibilité rationnelle* de G (FO[Rat]-théorie de G) est l'ensemble des FO[Rat]-énoncés satisfaits par G. Voir la proposition 3.3.32.

Chapitre 3

Etat de l'art

3.1 Langages

3.1.1 Grammaires, hiérarchie de Chomsky

Système de réécriture de mots

Un *système de réécriture de mots* P sur un alphabet Σ est un ensemble fini de couples $(l, r) \in \Sigma^*$. Un élément $(l, r) \in P$ est une *production* de P dont les membres gauche et droit sont respectivement l et r. La production (l, r) pourra être notée $l \to r$. La *relation de réécriture* $\underset{P}{\to}$ définie par P est :

$$\underset{P}{\to} := \{(ulv, urv) \mid (l, r) \in P \ \wedge \ u, v \in \Sigma^*\}$$

La *relation de dérivation* définie par P est la clôture réflexive et transitive de $\underset{P}{\to}$ pour la composition, notée $\underset{P}{\to}^*$.

Une production (l, r) est *croissante* si $|l| \leqslant |r|$

Grammaires

Une *grammaire* G est un quadruplet $G = (T, N, S, P)$, où
— T et N sont deux alphabets disjoints ;
— $S \in N$ est un symbole initial, appelé *axiome* de la grammaire G ;
— P est un système de réécriture de mots sur l'alphabet $T \cup N$ dont les membres gauches contiennent au moins un symbole appartenant à N.

Les symboles appartenant à T (respectivement N) sont dits *terminaux* (respectivement *non-terminaux*). Le langage $L(G)$ engendré par la grammaire G est l'ensemble des mots (terminaux) engendrés par G, c'est-à-dire l'ensemble des mots $u \in T^*$ qui peuvent être dérivés à partir de S selon P :

$$L(G) := \{u \in T^* \mid S \underset{P}{\to}^* u\}$$

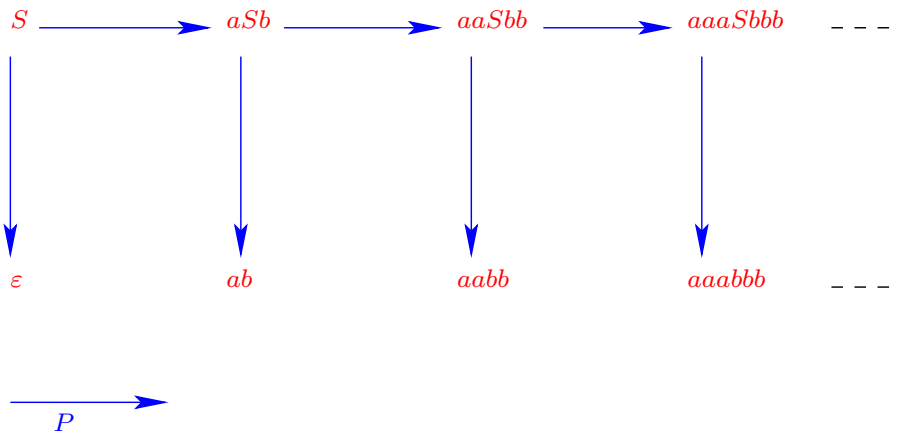


FIGURE 3.1 – Relation de réécriture d'une grammaire

Exemple 3.1.1. Considérons la grammaire G donnée par $T = \{a, b\}$, $N = \{S\}$ et $P = \{(S, \varepsilon), (S, aSb)\}$. Le langage de cette grammaire est $L(G) = \{a^n b^n \mid n \geqslant 0\}$. Voir la figure 3.1.

- Les langages engendrés par grammaire constituent la classe RE des langages *récursivement énumérables*.
- Les langages engendrés par les grammaire dont les productions sont croissantes, à l'exception possible de la production (S, ε) dont l'occurence dans P implique, cependant, que le non-terminal S n'apparaît dans aucune production appartenant à P, constituent la classe Cxt des langages *contextuels*.
- Les langages engendrés par les grammaires dont le membre gauche de toute production est un symbole non-terminal, constituent la classe Alg des langages *algébriques*.
- Les langages engendrés par les grammaires dont les productions sont de la forme (X, uY) ou bien (X, u), où $X, Y \in N$ et $u \in T^*$, constituent la classe Rat des langages rationnels.

Exemples de langages

Langage rationnel Les langages rationnels sont aussi les langages engendrés par les grammaires dont les productions sont de la forme (X, Yu) ou bien (X, u), où $X, Y \in N$ et $u \in T^*$.

Le langage $\{a^m b a^n \mid m, n \geqslant 0\}$ est le langage de la grammaire $(\{a, b\}, \{S, X\}, S, P)$, où P est le système de réécriture constitué des règles suivantes :

$$\begin{array}{lcl} S & \to & aS \\ S & \to & bX \\ X & \to & aX \\ X & \to & \varepsilon \end{array}$$

Langage algébrique Pour tout entier naturel n non nul, notons A_n l'alphabet défini par $A_n := \{a_1, \ldots, a_n, \overline{a_1}, \ldots, \overline{a_n}\}$. Le *langage de Dyck (sur n paires de parenthèses)* D_n est le langage de la grammaire $(A_n, \{S, X\}, S, P)$, où le système de réécriture P est constitué des productions suivantes :

$$\begin{array}{ccc} S & \to & SX \\ S & \to & \varepsilon \\ X & \to & a_1 S \overline{a_1} \\ & \cdots & \\ X & \to & a_n S \overline{a_n} \end{array}$$

Proposition 3.1.2. *Pour tout entier naturel n non nul, le langage de Dyck D_n est un langage algébrique, non rationnel.*

Le théorème suivant montre que les langages de Dyck constituent un générateur des langages algébriques.

Morphisme alphabétique Un morphisme alphabétique ϕ de Σ_1^* dans Σ_2^* est un morphisme de Σ_1^* dans Σ_2^* tel que $|\phi(a)| \leqslant 1$ pour tout $a \in \Sigma_1$.

Théorème 3.1.3 (Chomsky, Schützenberger)**.** *Un langage L est algébrique si et seulement s'il existe un entier naturel non nul n, un morphisme alphabétique ϕ et un langage rationnel K tel que $L = \phi(D_n \cap K)$.*

Langage contextuel Les langages $\{a^n b^n c^n \mid n \geqslant 0\}$, $\{a^{2^n} \mid n \geqslant 0\}$ ou encore $L = \{ww \mid w \in \Sigma^+\}$ sont des langages contextuels non algébriques. Le langage L est le langage de la grammaire dont l'ensemble des terminaux est Σ, l'ensemble des non terminaux est $\{X_a \mid a \in \Sigma\} \cup \{Y_a \mid a \in \Sigma\} \cup \{Z_a \mid a \in \Sigma\} \cup \{S\}$, le symbole non-terminal initial est S et l'ensemble des productions est l'union sur $(a, b) \in \Sigma^2$ de l'ensemble des règles suivantes :

$$\begin{array}{ccc} S & \to & aSX_a \\ S & \to & Y_a Z_a \\ Z_a X_b & \to & X_b Z_a \\ Y_a X_b & \to & Y_a Z_b \\ Z_a & \to & a \\ Y_a & \to & a \end{array}$$

Le mot *aabcaabc* appartient bien au langage de la grammaire définie ci-dessus :

$$S \to^* aabSX_bX_aX_a \to aabY_cZ_cX_bX_aX_a \to^* aabY_cX_bX_aX_aZ_c \to$$
$$aabY_cZ_bX_aX_aZ_c \to^* aabY_cX_aX_aZ_bZ_c \to^* aabY_cZ_aZ_aZ_bZ_c \to^* aabcaabc$$

Langage récursivement énumérable Rappelons d'abord que le problème de l'appartenance d'un mot à un langage contextuel est uniformément décidable. Ce rappel permet de fournir un exemple de langage récursivement énumérable non contextuel. En effet, soit $(G_i)_{i\in\mathbb{N}}$ une énumération des grammaires des langages contextuels et $(x_i)_{i\in\mathbb{N}}$ une énumération des Σ-mots. Le langage des Σ-mots $\{x_i \mid x_i \notin L(G_i)\}$ est un langage récursivement énumérable, non contextuel.

3.1.2 Propriétés de fermeture

Substitution Soient Σ_1 et Σ_2 deux alphabets. Une substitution σ de Σ_1 dans Σ_2 est une application de Σ_1^* dans $\mathcal{P}(\Sigma_2^*)$ telle que : $\sigma(\varepsilon) = \{\varepsilon\}$ et pour tous Σ_1-mots u et v, $\sigma(uv) = \sigma(u)\sigma(v)$. Il s'agit précisément d'un morphisme de Σ_1^* dans $\mathcal{P}(\Sigma_2^*)$.

L'image d'un langage L de Σ_1-mots par la substitution σ est le langage de Σ_2-mots $\sigma[L]$ défini par

$$\sigma[L] := \bigcup_{u\in L} \sigma(u)$$

Morphisme non effaçant Un morphisme de monoïdes $f : \Sigma_1^* \longrightarrow \Sigma_2^*$ est non effaçant si pour tout Σ_1-mot $u \neq \varepsilon$, $f(u)$ est distinct du mot vide ε.

Miroir Soit $w = a_1 \dots a_n$ un Σ-mot. Le mot miroir de w est $w^R := a_n \dots a_1$. Soit $L \subseteq \Sigma^*$ un langage de mots. Le langage miroir de L est l'ensemble des mots miroirs des mots de L.

Le tableau ci-dessous rappelle les différentes propriétés de fermeture de chacune des classes de langages définies par Chomsky.

fermée par	*RE*	*Cxt*	*Alg*	*Rat*
Union	oui	oui	oui	oui
Intersection	oui	oui	non	oui
Complémentation	non	oui	non	oui
Concaténation	oui	oui	oui	oui
Etoile	oui	oui	oui	oui
Intersection avec un langage rationnel	oui	oui	oui	oui
Miroir	oui	oui	oui	oui
Substitution	oui	non	oui	oui
Morphisme	oui	non	oui	oui
Morphisme non effaçant	oui	oui	oui	oui
Morphisme inverse	oui	oui	oui	oui

Remarque 3.1.4. Tout langage récursivement énumérable est image par un morphisme alphabétique (effaçant) d'un langage contextuel.

3.2 Automates (infinis)

3.2.1 Définitions

Un automate est la donnée d'un graphe ainsi que de sommets initiaux et finaux.

Fixons deux *couleurs* $\{\iota, o\}$. Soit Λ un alphabet. Un Λ-automate $\mathcal{A}$ est de la forme

$$G \ \cup \ \{\iota\} \times I_G \ \cup \ \{o\} \times F_G$$

où G est un Λ-graphe et I_G (respectivement F_G) l'ensemble des *sommets initiaux* de G (respectivement l'ensemble des *sommets finaux* de G), est une partie de V_G.

L'automate $\mathcal{A}$ est une structure relationnelle relativement à la signature relationnelle étendue $\{E_\lambda \mid \lambda \in \Lambda\} \cup \{\iota, o\}$. Son domaine et l'interprétation de ses symboles binaires sont ceux du graphe sous-jacent G et l'interprétation de ι (respectivement o) est la partie I_G (respectivement F_G).

L'automate $\mathcal{A}$ est un *automate déterministe* si son graphe sous-jacent G est déterministe et I_G est réduit à un singleton. On note alors i_G son unique sommet initial. Etant donnée une famille $\mathcal{F}$ d'automates, on note $\mathcal{F}_{\text{det}}$ la sous-famille de $\mathcal{F}$ constituée des automates déterministes.

Un chemin dans $\mathcal{A}$ est un chemin dans son graphe sous-jacent. Il est *acceptant* si sa source est un sommet initial et son but est un sommet final. Le *language accepté* par $\mathcal{A}$ est l'ensemble des étiquettes de ses chemins acceptants.

Exemple 3.2.1. Soit $L \subseteq \Sigma^*$ un langage de mots. L'automate des résidus de L défini par :

$$\{u^{-1}L \xrightarrow{a} (ua)^{-1}L \mid a \in \Sigma, u \in \Sigma^*\} \ \cup \ \{\iota\} \times \{L\} \ \cup \ \{o\} \times \{u^{-1}L \mid \varepsilon \in u^{-1}L\}$$

est un automate déterministe qui accepte L.

Dans les paragraphes 3.2.2 à 3.2.5, on donne pour chacune des classes de langages définies dans la hiérarchie de Chomsky, une ou plusieurs familles d'automates dont les langages acceptés sont exactement les langages de la classe considérée.

3.2.2 pour les langages réguliers

Proposition 3.2.2. *Les langages rationnels sont acceptés par les automates finis.*

Rappelons qu'un langage est rationnel si et seulement si l'ensemble de ses résidus à gauche est fini. En particulier, l'automate des résidus d'un langage accepté par un automate fini est fini. Cette propriété de fermeture par résiduation de la famille des automates acceptant les langages rationnels (*i.e.*, la famille des automates finis) ne sera plus vérifiée dans le cas des langages algébriques.

3.2.3 pour les langages algébriques

La machine abstraite automate à pile.

Considérons un système de réécriture P sur un alphabet Λ, vérifiant les conditions ci-dessous.

- L'alphabet Λ est l'union disjointe d'un alphabet Q, l'ensemble des *états*, et d'un alphabet $\Sigma = T \cup N$, où T est l'alphabet d'entrée et N l'alphabet de pile (observer que T et N ne sont pas nécessairement disjoints).
- Des éléments $q_i \in Q$, $\perp \in N$ et une partie $F \subseteq Q$ sont spécifiés ; il s'agit respectivement de l'*état initial*, du fond de pile ou lettre (de pile) de départ et de l'ensemble des *états finaux*.
- Les productions sont étiquetées et de la forme

$$zp \xrightarrow{a} uq \quad \text{avec } z \in N, p, q \in Q, u \in N^*, a \in T \cup \{\varepsilon\}$$

Un *automate à pile* est l'automate défini à partir d'un tel système de réécriture par

$$\{wzp \xrightarrow{a} wuq \mid zp \xrightarrow{a} uq \in P, w \in N^*\} \ \cup \ \{\iota\} \times \{\perp q_i\} \ \cup \ \{o\} \times N^*F$$

Un *automate à pile déterministe* est un automate à pile qui vérifie que pour tout $(z, p) \in N \times Q$,

1. soit zp est membre gauche d'une production étiquetée par ε, et dans ce cas zp n'est membre gauche d'aucune production étiquetée par $a \in T$,
2. soit zp n'est membre gauche d'aucune production étiquetée par ε, et dans ce cas pour chaque lettre $a \in T$, zp est membre gauche au plus d'une production étiquetée par $a \in T$.

Proposition 3.2.3. *Les automates à pile acceptent les langages algébriques.*

Automates suffixes de mots

Un *graphe de réécriture suffixe de mots* est une union finie de graphes de la forme

$$W(u \xrightarrow{a} v)$$

où W est un langage rationnel de mots, a est une étiquette, u, v sont des mots et

$$W(u \xrightarrow{a} v) = \{wu \xrightarrow{a} wv \mid w \in W\}$$

Un *automate suffixe de mots* est un automate de la forme

$$G \ \cup \ \{\iota\} \times I \ \cup \ \{o\} \times F$$

où G est un graphe de réécriture suffixe de mots, I et F sont des langages rationnels.

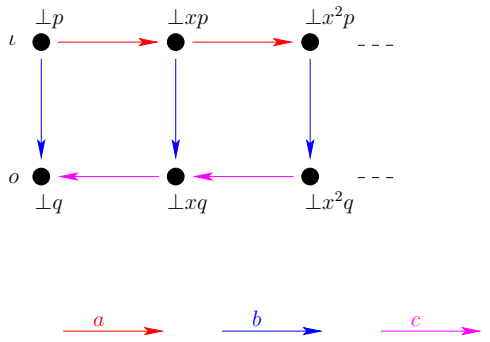


FIGURE 3.2 – Un automate suffixe de mots.

Proposition 3.2.4. *[10] Les automates suffixes de mots acceptent les langages algébriques.*

Exemple 3.2.5. Voir la figure 3.2 pour l'automate suffixe de mots H défini par

$$H := \perp x^*(p \xrightarrow{a} xp) \ \cup \ \perp x^*(p \xrightarrow{b} q) \ \cup \ \perp x^*(xq \xrightarrow{c} q) \ \cup \ \{\iota\} \times \{\perp p\} \ \cup \ \{o\} \times \{\perp q\}$$

Le langage accepté par H est $L(H) = \{a^n bc^n \mid n \geqslant 0\}$.

Exemple 3.2.6. L'automate des résidus du langage algébrique suivant :

$$L = a^+\{b^n c^n \mid n > 0\} \ \cup \ \{a^n b^n \mid n > 0\}d$$

n'est pas un automate suffixe de mots. Voir la figure 3.3.

3.2.4 pour les langages contextuels

Automates rationnels

Un Λ-graphe est *rationnel* s'il existe un alphabet Σ tel que pour tout $\lambda \in \Lambda_G$, l'ensemble des couples E^G_λ est une partie rationnelle du monoïde produit $\Sigma^* \times \Sigma^*$. En particulier, $V_G \subseteq \Sigma^*$. Un Λ-automate est rationnel si son graphe sous-jacent est rationnel et que l'ensemble de ses sommets initiaux (respectivement finaux) est un langage rationnel de Σ-mots. En fait, un graphe rationnel est un graphe dont chacune des relations E^G_λ ($\lambda \in \Lambda_G$) est reconnue par un transducteur fini.

Transducteur de mots

Soit Σ un alphabet. Un transducteur fini de Σ-mots est un $(\Sigma^* \times \Sigma^*)$-automate fini.

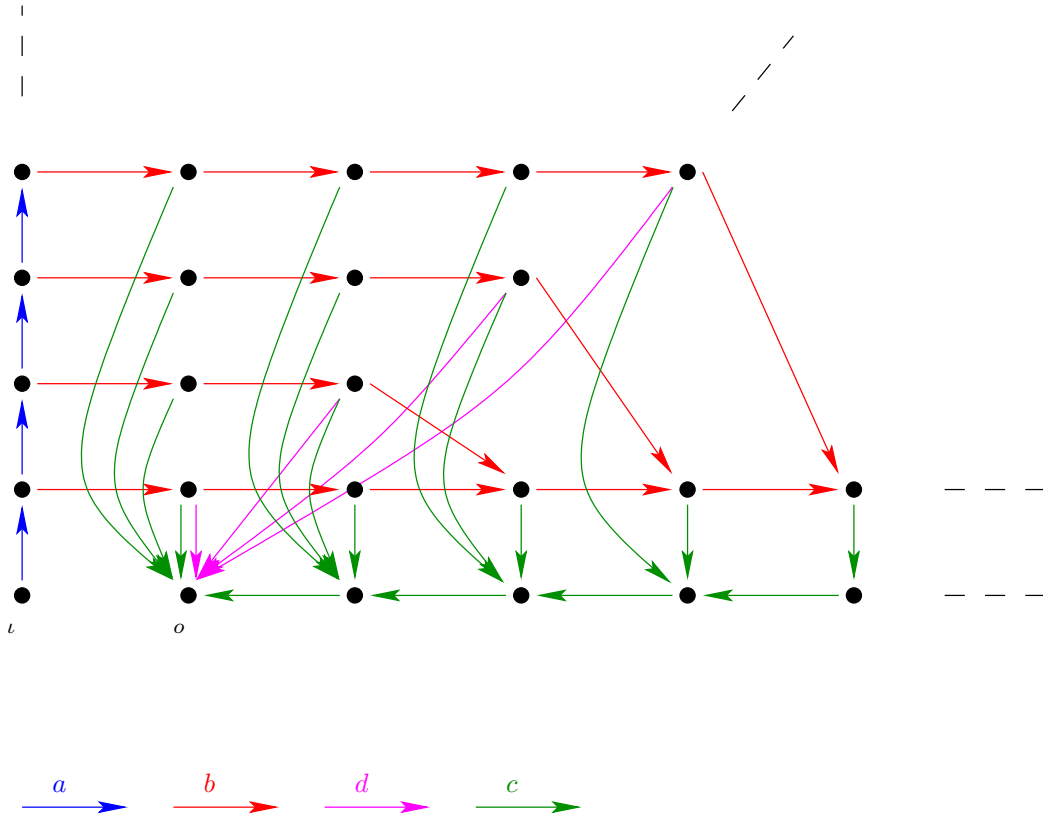


FIGURE 3.3 – L'automate des résidus d'un langage algébrique

Exemple 3.2.7. Soit $n > 0$ et $(u_0, u_1, \ldots, u_n)$ et $(v_0, v_1, \ldots, v_n)$ deux suites finies de Σ-mots de longueur n. Le transducteur T_{Post} suivant accepte les couples de Σ-mots de la forme $(u_0 u_{i_1} \ldots u_{i_m}, v_0 v_{i_1} \ldots v_{i_m})$, où $1 \leqslant i_1, \ldots, i_m \leqslant n$ et $m \geqslant 0$. Voir la figure 3.4.

$$T_{Post} := \{(p, (u_0, v_0), q)\} \cup \{(q, (u_i, v_i), q) \mid 1 \leqslant i \leqslant n\} \cup \{\iota\} \times \{p\} \cup \{o\} \times \{q\}$$

Nous reprendrons cet exemple pour montrer notamment que la théorie du premier ordre d'un graphe rationnel n'est pas décidable en général.

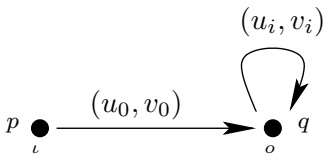


FIGURE 3.4 – Un transducteur de mots.

Théorème 3.2.8 ([32])**.** *Les langages contextuels sont acceptés par les automates rationnels.*

3.2.5 pour les langages récursivement énumérables

Machine de Turing étiquetée

Une machine de Turing étiquetée M est un uplet $(Q, \Sigma, P, \sharp, \Delta)$, où Q est un ensemble fini d'états, Σ est l'alphabet d'entrée que l'on suppose disjoint de Q, P est l'alphabet de la bande de travail, $\sharp \notin P$ est un symbole n'appartenant pas à P, Δ est un système de réécriture (étiqueté) dont les règles de réécriture sont de la forme

$$pA \xrightarrow{a} qB\delta$$

où $p, q \in Q$, $A, B \in P \cup \{\sharp\}$, $\delta \in \{+, -\}$ et $a \in \Sigma \cup \{\varepsilon\}$.

Posons $P_\sharp := P \cup \{\sharp\}$. Une configuration de M est de la forme $]u]p[v[$, où $p \in Q$ et $]u]$ (respectivement $[u[$) désigne le plus grand suffixe (respectivement préfixe) de u dont la première lettre (respectivement la dernière lettre) soit distincte de $\sharp$. Plus précisément, et par induction pour tout mot $u \in P_\sharp^*$

$$]\sharp u] := u \wedge]u] :=]u] \text{ si } u(1) \neq \sharp \quad \text{et} \quad [u\sharp[:= [u[\wedge [u[:= u \text{ si } u(|u|) \neq \sharp$$

Le graphe des configurations de M est

$$\begin{aligned} T(M) := \{]u]p[Av[\xrightarrow{a}]uA]q[v[\mid \ pA \xrightarrow[\Delta]{a} qB+, u, v \in P_\sharp^*\} \\ \cup \ \{]uC]p[Av[\xrightarrow{a}]u]q[CBv[\mid \ pA \xrightarrow[\Delta]{a} qB-, C \in P_\sharp, u, v \in P_\sharp^*\} \end{aligned}$$

Le graphe des configurations de M est un graphe avec ε-arcs.

Etant donné un ensemble rationnel de configurations $C \subseteq (Q \cup P_\sharp)^*$, l'automate des transitions de M restreint à C est

$$T(M, C) := \overline{T(M)}^{\varepsilon}_{|C} \ \cup \ \{\iota\} \times I \ \cup \ \{o\} \times F$$

où I (respectivement F) est un ensemble fini de configurations initiales (respectivement finales) de M.

Proposition 3.2.9 ([11])**.** *Les automates des transitions des machines de Turing acceptent les langages récursivement énumérables.*

3.2.6 Langages déterministes

Proposition 3.2.10. *Tout langage rationnel est reconnaissable par un automate fini déterministe.*

Il suffit pour un langage rationnel donné de considérer son automate des résidus.

Considérons le cas des langages algébriques. Un langage algébrique est déterministe s'il est accepté par un automate à pile déterministe.

Proposition 3.2.11. *La classe des langages algébriques déterministes est strictement incluse dans celle des langages algébriques.*

Rappelons quelques propriétés de fermeture des langages algébriques déterministes.

Proposition 3.2.12. *La classe des langages algébriques déterministes est fermée par complémentation, morphisme inverse et résidu à droite par un langage rationnel.*

Cependant, la classe des langages déterministes algébriques n'est pas fermée par union, intersection, concaténation, étoile et miroir.

Dans le cas des langages contextuels, une notion de machine linéairement bornée non déterministe (non rappelée dans ce manuscrit) a été définie par Kuroda dans [23]. Les langages contextuels sont précisément les langages reconnus par les machines linéairement bornées [23]. En fait, la notion de machine linéairement bornée déterministe a été initialement définie par Myhill dans [34]. C'est une question encore ouverte, posée par Kuroda dans [23], que de déterminer si tout langage contextuel est reconnaissable par une machine linéairement bornée déterministe.

Enfin, mentionnons que toute machine de Turing étiquetée peut être transformée en une machine de Turing étiquetée déterministe. Tout langage récursivement énumérable est ainsi reconnaissable par un automate déterministe.

3.2.7 L'algèbre de Boole des langages algébriques visibles

Dans ce paragraphe, nous rappelons divers résultats issus de [1] concernant les langages d'automates à pile visibles.

Soit (T_{-1}, T_0, T_1) un triplet d'alphabets disjoints. Un tel triplet sera appelé *alphabet visible*. Pour définir la notion d'automate à pile visible sur (T_{-1}, T_0, T_1), nous reprenons les notations introduites dans le paragraphe 3.2.3 à propos des automates à pile.

Un *automate à pile visible* sur (T_{-1}, T_0, T_1) est automate à pile, défini à partir d'un système de réécriture dont l'alphabet d'entrée T est $T_{-1} \cup T_0 \cup T_1$ et dont les productions ne sont pas étiquetées par ε et satisfont les conditions suivantes :

- les productions étiquetées par $a \in T_1$ sont de la forme : $zp \xrightarrow{a} yxq$
- les productions étiquetées par $a \in T_0$ sont de la forme : $zp \xrightarrow{a} yq$
- les productions étiquetées par $a \in T_{-1}$ sont de la forme : $zp \xrightarrow{a} q$ ou bien $\bot p \xrightarrow{a} \bot q$

où $x, y, z \in N$, $p, q \in Q$

Un automate à pile visible est déterministe si son graphe des configurations l'est.

Un *langage d'automate à pile visible* sur (T_{-1}, T_0, T_1) est un langage accepté par un automate à pile visible sur (T_{-1}, T_0, T_1).

Soit (T^1_{-1}, T^1_0, T^1_1) et (T^2_{-1}, T^2_0, T^2_1) deux alphabets visibles. Un *renommage* de (T^1_{-1}, T^1_0, T^1_1) vers (T^2_{-1}, T^2_0, T^2_1) est un morphisme

$$f : (T^1_{-1} \cup T^1_0 \cup T^1_1)^* \longrightarrow (T^2_{-1} \cup T^2_0 \cup T^2_1)^*$$

tel que $f[T_{-1}^1] \subseteq f[T_{-1}^2]$, $f[T_0^1] \subseteq f[T_0^2]$ et $f[T_1^1] \subseteq f[T_1^2]$.

Théorème 3.2.13. *[1] La classe des langages d'automates à pile visibles sur* (T_{-1}, T_0, T_1) *est fermée par union, intersection, concaténation, étoile de Kleene, complémentation et renommage.*

La fermeture par complémentation repose sur le fait que tout langage d'automate à pile visible est accepté par un automate à pile visible déterministe.

Caractérisation logique des langages d'automates à pile visibles relativement à un alphabet visible

Rappelons d'abord que tout mot $w = w(1) \dots w(n)$ sur un alphabet Σ est une structure relationnelle (que l'on s'autorise encore à noter w) relativement à la signature relationnelle $\{E_a \mid a \in \Sigma\} \cup \{\leqslant\}$, où $\leqslant$ est un symbole binaire et l'arité de tout E_a ($a \in \Sigma$) est 1. Le domaine de w est $\{1, \dots, n\}$, l'interprétation de tout symbole E_a ($a \in \Sigma$) est ${E_a}^w = \{i \mid 1 \leqslant i \leqslant n, w_i = a\}$ et l'interprétation du symbole binaire $\leqslant$ est $\leqslant^w = \{(i,j) \mid 1 \leqslant i \leqslant j \leqslant n\}$.

Rappelons que :

Théorème 3.2.14 (Büchi)**.** *Les langages rationnels de* Σ*-mots sont les langages* $\mathrm{MSO}[\{E_a \mid a \in \Sigma\} \cup \{\leqslant\}]$*-définissables.*

Les langages d'automates à pile visibles peuvent être caractérisés d'un point de vue de la logique. Nous allons expliciter cette caractérisation. Pour ce faire, on enrichit tout mot sur un alphabet visible d'une relation d'imbrication.

Soit $w = w(1) \dots w(n)$ un mot sur (T_{-1}, T_0, T_1). Une position $1 \leqslant i \leqslant n$ du mot w est une position d'*appel* (respectivement de *retour*) si $w_i \in T_1$ (respectivement $w_i \in T_{-1}$). La position i est une position *interne* si i n'est ni une position d'appel, ni une position de retour, c'est-à-dire $w_i \in T_0$.

La *relation d'imbrication* de w est l'ensemble des couples (i,j) tels que j est une position de retour de w et i est la plus grande position d'appel strictement inférieure à j. Le mot w est alors une structure relationnelle relativement à la signature

$$\sigma_{(T_{-1},T_0,T_1)} := \{E_a \mid a \in T_{-1} \cup T_0 \cup T_1\} \cup \{\leqslant, \mu\}$$

où μ est un symbole binaire dont l'interprétation μ^w est la relation d'imbrication.

Théorème 3.2.15. *[1] Les langages d'automates à pile visibles relativement à l'alphabet visible* (T_{-1}, T_0, T_1) *sont les langages* $\mathrm{MSO}[\sigma_{(T_{-1},T_0,T_1)}]$*-définissables.*

Exemple 3.2.16. Considérons l'alphabet visible $(\{c\}, \{b\}, \{a\})$. La formule logique ci-dessous exprime que toute position d'appel i admet une position de retour j correspondante, c'est-à-dire telle que (i,j) appartienne à la relation d'imbrication.

$$\forall x \ (a(x) \ \longrightarrow \ \exists y \ (c(y) \ \wedge \ \mu(x,y)))$$

Exemple 3.2.17. Le langage $\{a^n b a^n \mid n \geqslant 0\}$, bien qu'algébrique, n'est pas un langage d'automate à pile visible quels que soient les choix possibles pour a et b en termes de position d'appel, de retour ou interne.

3.2.8 Langages des réseaux de Petri

Dans la littérature, différents types de langages de réseaux de Petri on été considérés [16, 20, 38, 47], selon que la fonction d'étiquetage est injective ou non, ou encore qu'elle permet des ε-transitions (*i.e.*, des transitions étiquetées par le mot vide ε) ou non, ou encore que l'ensemble des sommets finaux est fini ou bien égal à l'ensemble des sommets accessibles. En général, les différentes investigations se concentrent sur les propriétés de fermeture de ces langages [20] et sur leurs relations par rapport aux autres familles de langages classiques apparaissant dans la hiérarchie de Chomsky [47, 42, 24]. Par ailleurs, il est aussi bien connu que la régularité (respectivement l'algébricité) des langages de systèmes d'addition de vecteurs (un modèle équivalent à celui des réseaux de Petri d'un point de vue de l'accessibilité) dont la fonction d'étiquetage est injective et n'autorise pas d'ε-transition est décidable [47] (respectivement [42, 24]). Enfin, notons que si certains types de langages de réseaux de Petri sont fermés par union et par intersection, la complémentation reste problématique.

Dans ce paragraphe, la fonction d'étiquetage des réseaux de Petri considérés n'est pas nécessairement injective mais n'autorise pas d'ε-transition. On rappelle alors une caractérisation logique des langages acceptés par ces réseaux de Petri (voir le théorème 3.2.20). Ceux-ci ne forment pas une algèbre de Boole. Dans le chapitre 6 de ce manuscrit, on considérera une notion de réseau de Petri généralisé dont les langages acceptés forment une algèbre de Boole.

Etant donnés deux vecteurs d'entiers naturels v_1 et v_2, de même dimension $p \in \mathbb{N}$, on écrira $v_1 \leqslant v_2$ si $v_1(i) \leqslant v_2(i)$ pour tout $i \in \{1, \ldots, p\}$.

Soit A un ensemble fini de couples de vecteurs d'entiers naturels de même dimension $p > 1$ et $l : A \to \Sigma$ une fonction d'étiquetage de ces couples par des lettres appartenant à un alphabet Σ.

Le *réseau de Petri étiqueté* de dimension p défini à partir du couple (A, l) est le graphe G_A sur l'ensemble des vecteurs d'entiers naturels de dimension p, défini par

$$G_A := \{v \xrightarrow{l((a_1,a_2))} v - a_1 + a_2 \mid v \in \mathbb{N}^p, a_1 \leqslant v, (a_1, a_2) \in A\}$$

Un *langage de réseau de Petri* est un langage accepté par un automate dont le graphe sous-jacent est un réseau de Petri et qui possède un unique sommet initial et un ensemble fini d'états finaux.

Exemple 3.2.18. Considérons le réseau de Petri étiqueté défini à partir de :

$$A := \{(\overbrace{\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}1\\0\\1\\0\end{pmatrix}}^{a}), (\overbrace{\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\1\\0\end{pmatrix}}^{a}), (\overbrace{\begin{pmatrix}0\\1\\1\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\1\end{pmatrix}}^{b}), (\overbrace{\begin{pmatrix}0\\1\\0\\1\end{pmatrix}, \begin{pmatrix}0\\0\\0\\1\end{pmatrix}}^{c}), (\overbrace{\begin{pmatrix}0\\0\\0\\1\end{pmatrix}, \begin{pmatrix}0\\0\\0\\1\end{pmatrix}}^{c})\}$$

Muni du sommet initial $\begin{pmatrix}1\\0\\0\\0\\0\end{pmatrix}$ et du sommet final $\begin{pmatrix}0\\0\\0\\0\\1\end{pmatrix}$, il accepte le langage

$$\{a^n b^n c^n \mid n > 0\}$$

Voir la figure 3.5.

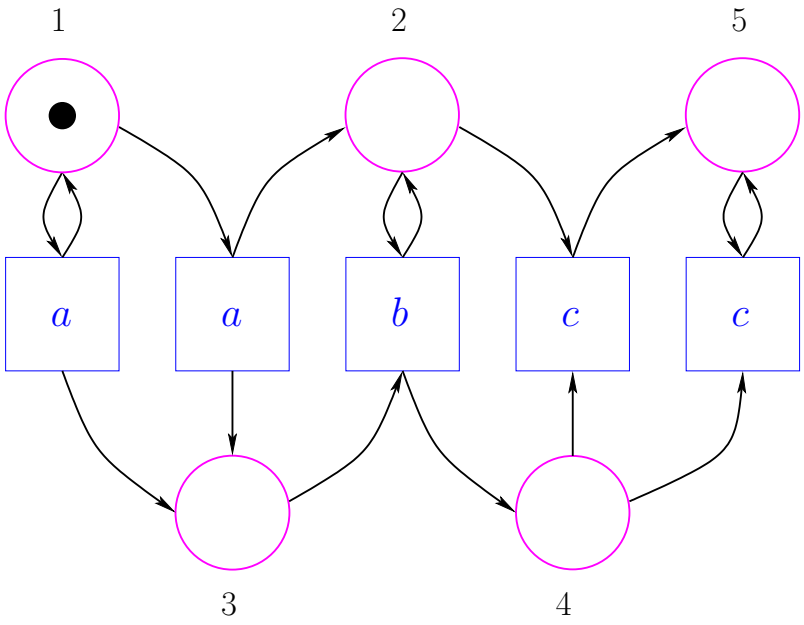


FIGURE 3.5 – Un réseau de Petri.

Pour donner une caractérisation logique des langages de réseaux de Petri, on commence par étendre la signature des Σ-mots par deux nouveaux *symboles binaires relationnels du second-ordre* $\leqslant_g$ et $=_g$ dont on va préciser l'interprétation.

Plus précisément, dans ce paragraphe, une signature relationnelle σ est un ensemble de symboles de relation du premier ordre ou bien du second ordre. Chaque symbole de relation (du premier ordre ou du second ordre) possède une arité qui est un entier naturel. Une structure relationnelle $\mathfrak{A}$ relativement à cette notion de signature relationnelle étendue est obtenue comme précédemment mais en ajoutant l'interprétation des symboles relationnels du second ordre, c'est-à-dire en spécifiant pour chaque symbole relationnel du second ordre S, une relation $a(S)$-aire sur $\mathcal{P}(A)$, où A est le domaine de $\mathfrak{A}$. On note $S^{\mathfrak{A}}$ l'interprétation de S. Par conséquent, $S^{\mathfrak{A}} \subseteq \mathcal{P}(A)^{a(S)}$.

Précisons à présent ce que sont les formules de la logique du second ordre monadique relativement à σ (MSO[σ]-formules). Les formules atomiques sont de la forme $x \in X$ ou bien $R(x_1, \ldots x_{a(R)})$ ou bien $S(X_1, \ldots X_{a(S)})$ où R (respectivement S) est un symbole relationnel du premier ordre (respectivement un symbole relationnel du second ordre) et où X_i ($i \in \{1, \ldots, a(S)\}$) peut aussi bien être un symbole

de variable du second ordre monadique ou un symbole relationnel du second ordre unaire. Rappelons, en effet, que l'interprétation dans une structure relationnelle d'un tel symbole est une partie du domaine de la structure considérée.

1. Les formules atomiques sont des formules.
2. Si ϕ et ψ sont des formules, alors $(\phi \wedge \psi)$ et $\neg\phi$ sont des formules.
3. Si ϕ est une formule, alors $\exists x\phi$ est une formule, où x est une variable du premier ordre.
4. Si ϕ est une formule, alors $\exists X\phi$ est une formule.

Une MSO[σ]-formule est dite du premier ordre si elle est obtenue en n'appliquant que les trois premières règles ci-dessus.

Un Σ-mot $w = w_1 \dots w_n$ $(n \geqslant 0)$ est une structure relationnelle relativement à la signature relationnelle σ_g constituée

— des symboles du premier ordre $\{E_a \mid a \in \Sigma\} \cup \{\leqslant\}$ dont l'interprétation a déjà été précisée précédemment,

— du symbole relationnel du second ordre $\leqslant_g$, dont l'interprétation $\leqslant_g^w$ est l'ensemble des couples $(P, Q) \in \mathcal{P}(\{1, \dots, n\})^2$ de parties de $\{1, \dots, n\}$ telles que pour tout entier naturel $0 \leqslant m \leqslant n$

$$\mathrm{card}(P \cap \{1, \dots, m\}) \leqslant \mathrm{card}(Q \cap \{1, \dots, m\})$$

— du symbole relationnel du second ordre $=_g$, dont l'interprétation $=_g^w$ est l'ensemble des couples $(P, Q) \in \mathcal{P}(\{1, \dots, n\})^2$ de parties de $\{1, \dots, n\}$ telles que

$$(P, Q) \in \leqslant_g^w \text{ et } \mathrm{card}(P) = \mathrm{card}(Q)$$

Ainsi $\sigma_g := \{E_a \mid a \in \Sigma\} \cup \{\leqslant\} \cup \{\leqslant_g, =_g\}$.

Exemple 3.2.19. Considérons $A_1 = \{a_1, \overline{a_1}\}$. Un A_1-mot $w = w_1 \dots w_n$ appartient au langage de Dyck D_1 si et seulement si $\{i \leqslant n \mid w_i = \overline{a_1}\} =_g \{i \leqslant n \mid w_i = a_1\}$.

Théorème 3.2.20 ([36])**.** *Les Σ-langages de réseaux de Petri sont les langages définissables par un* MSO[σ_g]*-énoncé de la forme $\exists X_1 \dots \exists X_n \phi$, où $n \geqslant 0$ et ϕ est une formule du premier ordre.*

Corollaire 3.2.21. *La classe des langages des réseaux de Petri est stable par intersection et par union.*

Donnons quelques exemples de langages de réseaux de Petri ainsi que les formules qui les caractérisent. Pour ce faire, on s'autorisera, comme à l'accoutumée, à faire usage des symboles logiques supplémentaires $\longrightarrow$, $\longleftrightarrow$, $\vee$. On s'autorisera aussi le raccourci $x = y$ pour $x \leqslant y \ \wedge \ y \leqslant x$ et $x < y$ pour $(x \leqslant y \ \wedge \ \neg\, x = y)$.

Exemple 3.2.22. Le langage $L = \{a^n b^n \mid n > 0\}$ est un langage de réseau de Petri défini par le MSO-énoncé suivant

$$(E_b =_g E_a \ \wedge \ \forall x \, \forall y \, ((E_a(x) \ \wedge \ E_b(y)) \longrightarrow \ x < y))$$

Exemple 3.2.23. Le langage contextuel $L = \{a^n b^{2n} c^n \mid n > 1\}$ est un langage de réseau de Petri défini par le MSO-énoncé suivant

$$\exists B_1\, \exists B_2\, (B_1 =_g E_a \wedge E_c =_g B_2 \wedge B_2 =_g B_1 \wedge \forall x\, (E_b(x) \longleftrightarrow (x \in B_1 \vee x \in B_2))$$
$$\wedge\, \forall u\, \forall x\, \forall y\, \forall z\, ((E_a(u) \wedge x \in X \wedge y \in Y \wedge E_c(z)) \longrightarrow u < x \wedge x < y \wedge y < z))$$

Plus généralement :

Proposition 3.2.24 ([39])**.** *Les langages des réseaux de Petri sont contextuels.*

Exemple 3.2.25. Le langage $L = \{w \in \Sigma^+ \mid w \neq w^R\}$, où w^R est le mot miroir de w, est un langage de réseau de Petri défini par le MSO-énoncé suivant

$$\exists X\, \exists Y\, \exists x\, \exists y$$
$$(\forall z\, ((z \in X \longleftrightarrow z \leqslant x) \wedge (z \in Y \leftrightarrow y \leqslant z)) \wedge Y =_g X \wedge \bigvee_{a \in \Sigma} E_a(x) \wedge \neg\, E_a(y))$$

Proposition 3.2.26 ([22])**.** *Le langage algébrique des palindromes $L = \{w \in \Sigma^+ \mid w = w^R\}$ n'est pas un langage de réseau de Petri.*

Propriétés de fermeture des langages de réseaux de Petri

La proposition 3.2.26 et l'exemple 3.2.25 montrent que la classe C_{Petri} des langages des réseaux de Petri n'est pas stable par complémentation.

Plus généralement, le tableau ci-dessous rappelle quelques propriétés de fermeture de cette classe de langages [39].

fermée par	C_{Petri}
Union	oui
Intersection	oui
Complémentation	non
Concaténation	oui
Etoile	non
Intersection avec un langage rationnel	oui
Miroir	oui
Substitution	non
Substitution finie	oui
Morphisme	oui
Morphisme inverse	oui

3.3 Transformations de graphes et logiques

Un problème important en vérification automatique consiste à déterminer (ou à étendre) des classes de graphes infinis dont la théorie relativement à une logique donnée est décidable.

Une première approche consiste à considérer de judicieuses transformations de graphes, comme par exemple la transformation de structure arborescente (qui préserve la décidabilité de la logique du second ordre monadique) ou bien les interprétations logiques.

3.3.1 Structure arborescente et dépliage

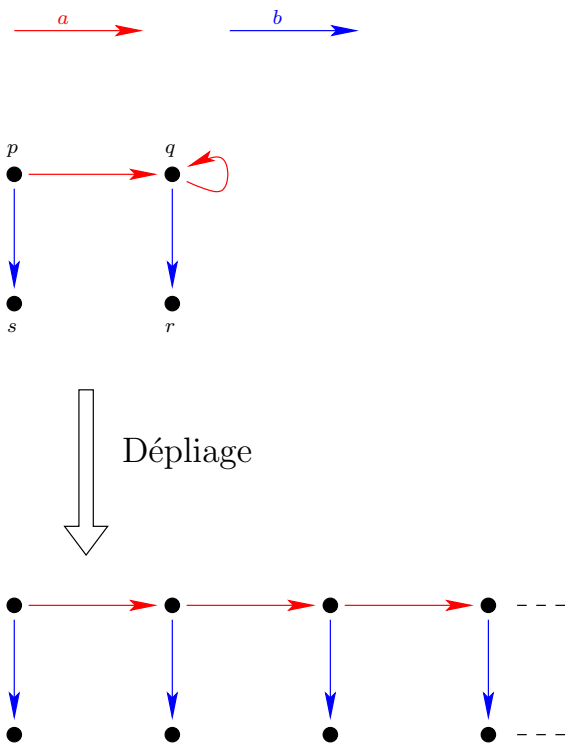


FIGURE 3.6 – Dépliage.

Définition 3.3.1 (Dépliage)**.** Soit G un graphe et $r \in V_G$ l'un de ses sommets. Le déplié Unfold(G, r) de G à partir de r est le graphe dont les arcs sont de la forme

$$U \xrightarrow{\lambda} U(v, \lambda, w)$$

où U est un chemin de source r et de but v dans G et $v \xrightarrow[G]{\lambda} w$.

Exemple 3.3.2. Soit $G_0 := \{(p, a, q), (q, a, q), (q, b, r), (p, b, s)\}$. Voir la figure 3.6 pour le déplié de G_0 à partir de p.

Théorème 3.3.3 ([15]). *Le dépliage depuis un sommet* MSO*-définissable préserve la décidabilité de la logique du second ordre monadique.*

Une autre transformation de graphes qui préserve la décidabilité de la logique du second ordre monadique est celle de structure arborescente, définie par Muchnick [43].

Définition 3.3.4 (Structure arborescente)**.** Soit G un graphe et $\sharp \in \Lambda \setminus \Lambda_G$. La structure arborescente de G par le symbole $\sharp$, notée $\text{Treegraph}_\sharp(G)$, est définie par :

$$\text{Treegraph}_\sharp(G) := \{(wu, \lambda, wv) \mid w \in V_G^* \ \wedge \ (u, \lambda, v) \in G\} \\ \cup \ \{(wu, \sharp, wuu) \mid w \in V_G^*, u \in V_G\}$$

Voir la figure 3.7 pour un exemple de structure arborescente.

Théorème 3.3.5 ([48])**.** *La transformation de structure arborescente préserve la décidabilité de la logique du second ordre monadique.*

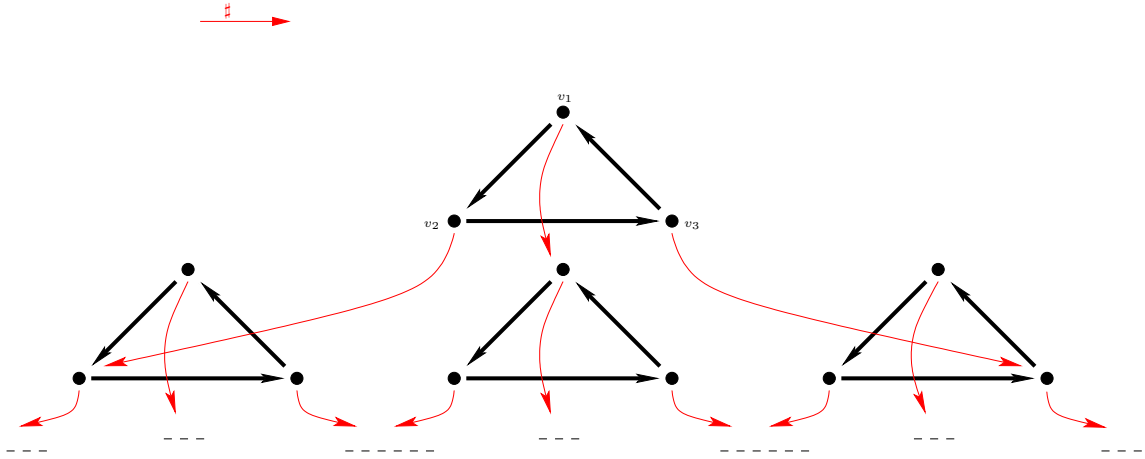


FIGURE 3.7 – La structure arborescente du triangle.

3.3.2 Interprétations logiques et substitutions

Soit $\Sigma_1, \Sigma_2 \subseteq \Lambda$ deux ensembles d'étiquettes et L une « logique ».

Définition 3.3.6 (L-interprétation logique)**.** Une L-interprétation $\mathcal{I}$ de Σ_1 vers Σ_2 est la donnée d'une $L[\{E_a \mid a \in \Sigma_1\}]$-formule $\delta(x)$, et pour chaque $\lambda \in \Sigma_2$, d'une $L[\{E_a \mid a \in \Sigma_1\}]$-formule $\phi_\lambda(x, y)$, où x et y sont des variables du premier ordre.

Soit G un Σ_1-graphe. L'interprétation $\mathcal{I}$ appliquée à G est le Σ_2-graphe $\mathcal{I}(G)$ défini par :

$$\mathcal{I}(G) = \{v \xrightarrow{\lambda} w \mid G \models \phi_\lambda(v,w) \ \wedge \ \delta(v) \ \wedge \ \delta(w)\}$$

Exemple 3.3.7. L'ensemble des formules suivantes constitue une MSO-interprétation.

- $\delta(x) = \ (x = x)$
- $\phi_a(x,y) = \ x \xrightarrow{a} y$
- $\phi_b(x,y) = \ x \xrightarrow{b} y$
- $\phi_c(x,y) = \ \exists z \ \exists t \ z \xrightarrow{a^+} t \ \wedge \ z \xrightarrow{b} y \ \wedge \ t \xrightarrow{b} x$

où $z \xrightarrow{a^+} t$ est un raccourci d'écriture pour la formule aChemin(z,t) explicitée ci-après et qui traduit le fait que z et t sont respectivement source et but d'un chemin étiqueté par un mot non vide appartenant à a^*. D'abord, observons que la formule intermédiaire

$$\text{aClose}(X) = \forall x \ \forall y \ ((x \in X \ \wedge x \xrightarrow{a} y) \ \longrightarrow \ y \in X)$$

exprime intuitivement que la partie X est fermée par les arcs étiquetés par a. Ensuite, nous pouvons expliciter la formule aChemin(z,t) dont $z \xrightarrow{a^+} t$ est un raccourci :

$$\text{aChemin}(z,t) = \exists x \ (z \xrightarrow{a} x \ \wedge \ \forall X \ ((x \in X \ \wedge \ \text{aClose}(X)) \ \longrightarrow \ t \in X))$$

Voir la figure 3.8 pour l'application de cette MSO-interprétation à Unfold(G_0, p).

Proposition 3.3.8. *Toute* MSO*-interprétation préserve la décidabilité de la logique du second ordre monadique.*

Transduction monadique

Observons que l'ensemble des sommets d'un graphe obtenu par interprétation logique est inclus dans l'ensemble des sommets du graphe dont il est interprétation : une interprétation logique ne permet pas d'augmenter le domaine du graphe sur lequel elle s'applique. Pour dépasser cette restriction, on peut considérer la notion de transduction monadique.

Soient K et Σ deux alphabets disjoints.

La *transformation de* (K, Σ)*-copie* s'applique aux Σ-graphes. A un Σ-graphe G, elle associe le $(\Sigma \cup K)$-graphe G' défini par :

$$G' = G \ \cup \ \{v \xrightarrow{k} (v,k) \mid v \in V_G, k \in K\}$$

Définition 3.3.9 (Transduction monadique)**.** Une transduction monadique $\mathcal{T} = (K, \mathcal{I})$ de Σ_1 vers Σ_2 est une transformation de (K, Σ_1)-copie suivie d'une MSO-interprétation $\mathcal{I}$ de $\Sigma_1 \cup K$ vers Σ_2.

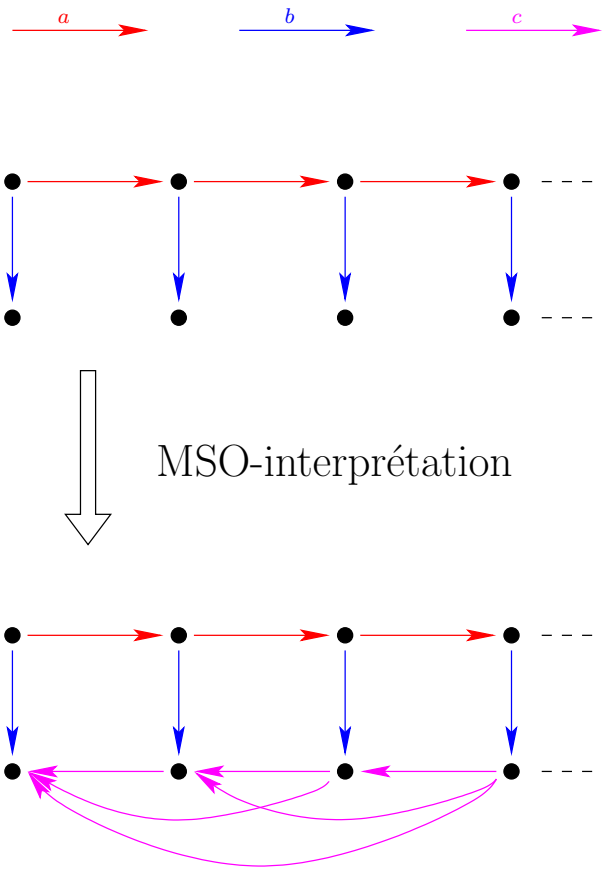


FIGURE 3.8 – Une MSO-interprétation de Unfold(G_0, p).

Proposition 3.3.10. *La transformation de transduction monadique préserve la décidabilité de la logique du second ordre monadique.*

Les transformations de structure arborescente et de transduction monadique sont les principales transformations préservant la décidabilité de la logique du second ordre monadique.

Substitution inverse

Nous allons maintenant considérer une autre transformation qui s'appuie sur la notion de substitution.

Soit G un Σ_1-graphe. Notons $\overline{\Sigma_1}$ un ensemble d'étiquettes disjoint de Σ_1 et en bijection avec celui-ci. L'image de $\lambda \in \Sigma_1$ est notée $\overline{\lambda}$. Le graphe $\widetilde{G}$ est alors obtenu à partir de G en ajoutant les arcs inverses :

$$\widetilde{G} = G \ \cup \ \{v \xrightarrow{\overline{\lambda}} u \mid u \xrightarrow[G]{\lambda} v\}$$

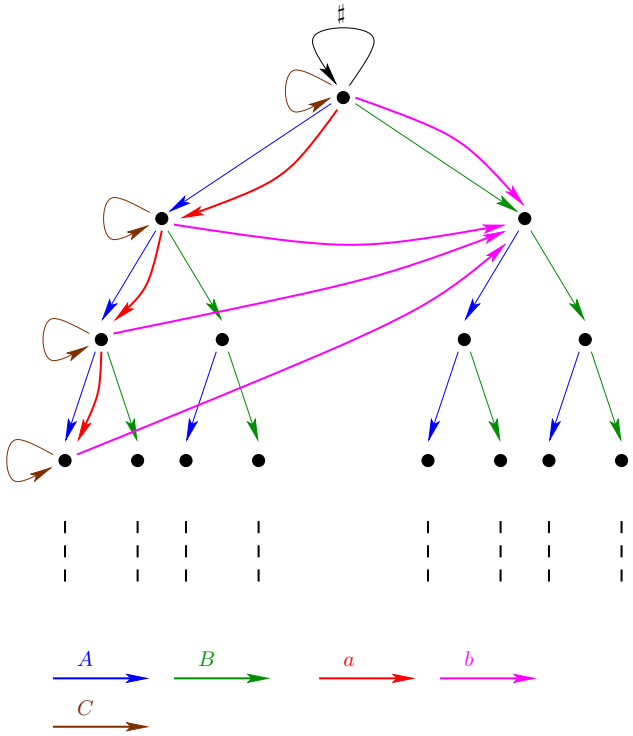


FIGURE 3.9 – Une substitution rationnelle appliquée à un arbre binaire infini marqué.

Une substitution h de Σ_2 vers $\Sigma_1 \cup \overline{\Sigma_1}$, appliquée à G de manière inverse, est le Σ_2-graphe

$$h^{-1}(G) := \{u \xrightarrow{\lambda} v \mid u \xrightarrow[\widetilde{G}]{h(\lambda)} v\}$$

Une substitution de Σ_1 vers Σ_2 est dite *rationnelle* (respectivement *finie*) si pour tout $\lambda \in \Sigma_1$, $h(\lambda)$ est un langage rationnel (respectivement fini).

Exemple 3.3.11. (Substitution rationnelle inverse) On applique la substitution rationnelle définie par $h(a) = CA$ et $h(b) = C\overline{A}^*\sharp B$ à l'arbre binaire infini marqué suivant :

$$\{u \xrightarrow{A} uA \mid u \in \{A, B\}^*\} \cup \{u \xrightarrow{B} uB \mid u \in \{A, B\}^*\} \cup \{\varepsilon \xrightarrow{\sharp} \varepsilon\} \cup \{u \xrightarrow{C} u \mid u \in A^*\}$$

Voir la figure 3.9.

Exemple 3.3.12. (Substitution rationnelle inverse) On applique la substitution rationnelle définie par $h(c) = \sharp A + \sharp\overline{A}^*\sharp\overline{B}AB$ à l'arbre binaire infini marqué suivant :

$$\{u \xrightarrow{A} uA \mid u \in \{A, B\}^*\} \cup \{u \xrightarrow{B} uB \mid u \in \{A, B\}^*\} \cup \{u \xrightarrow{\sharp} u \mid u \in A^*BA^*\}$$

Voir la figure 3.10 pour le graphe obtenu.

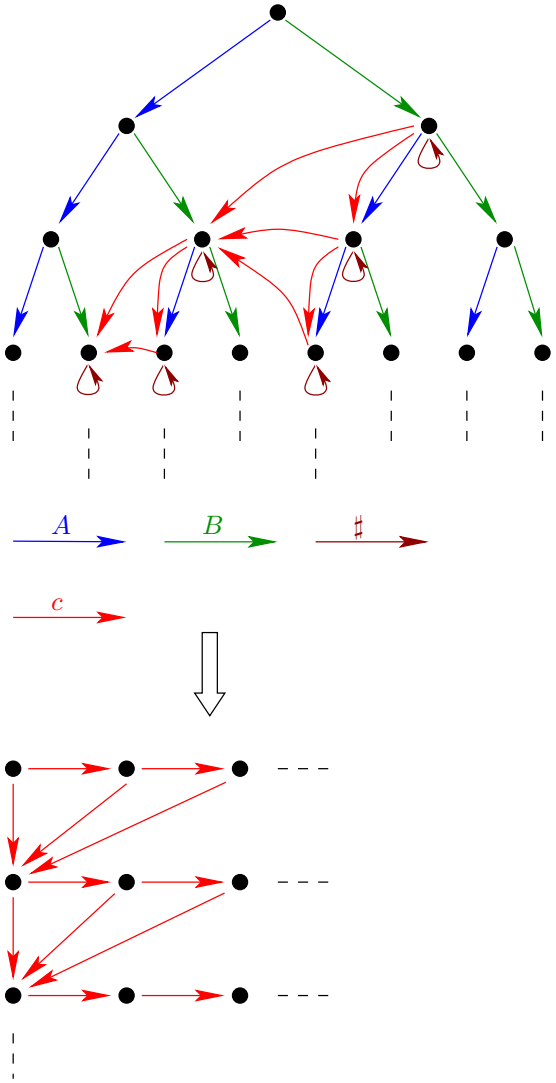


FIGURE 3.10 – Une substitution rationnelle appliquée à un arbre binaire infini marqué.

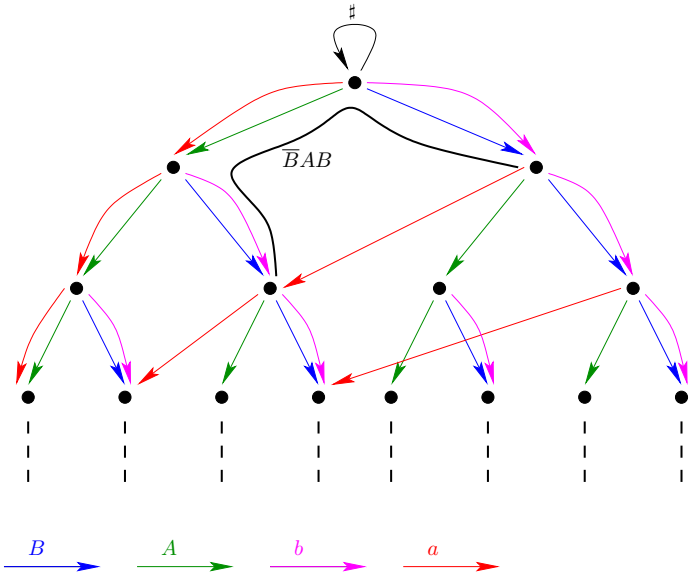


FIGURE 3.11 – Le quart de la grille infinie comme substitution inverse de l'arbre binaire.

Exemple 3.3.13. Soit h la substitution définie par $h(a) = \{\overline{B}^m \sharp AB^m \mid m \geqslant 0\}$ et $h(b) = B$. La substitution h appliquée à l'arbre binaire infini marqué

$$\{u \xrightarrow{A} uA \mid u \in \{A, B\}^*\} \cup \{u \xrightarrow{B} uB \mid u \in \{A, B\}^*\} \cup \{\varepsilon \xrightarrow{\sharp} \varepsilon\}$$

fournit le quart de la grille infinie sur $\{a, b\}$. Observons que h n'est pas une substitution rationnelle. Voir les figures 3.11 et 3.12.

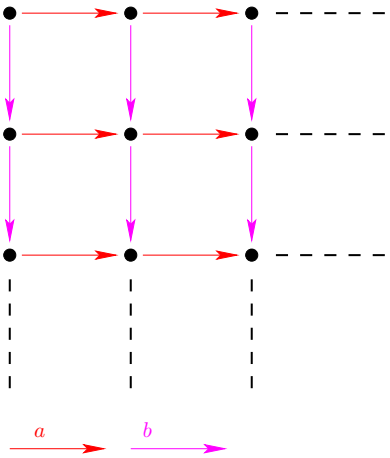


FIGURE 3.12 – Le quart de la grille infinie.

Une substitution rationnelle inverse est un cas particulier de MSO-interprétation, et on peut appliquer la proposition 3.3.8.

Proposition 3.3.14. *Les substitutions rationnelles inverses préservent la décidabilité de la logique du second ordre monadique.*

3.3.3 Graphe des parties

Définition 3.3.15 (Graphe des parties d'un graphe)**.** Soit G un graphe. Le graphe Parties(G) des parties de G est

$$\text{Parties}(G) := \{P \xrightarrow{\subseteq} Q \mid P \subseteq Q \subseteq V_G\} \ \cup \ \{\{v\} \xrightarrow{\lambda} \{w\} \mid v, w \in V_G, v \xrightarrow[G]{\lambda} w\}$$

Le *graphe des parties finies* de G est la restriction de Parties(G) à $\wp(V_G)$.
Voir la figure 3.13

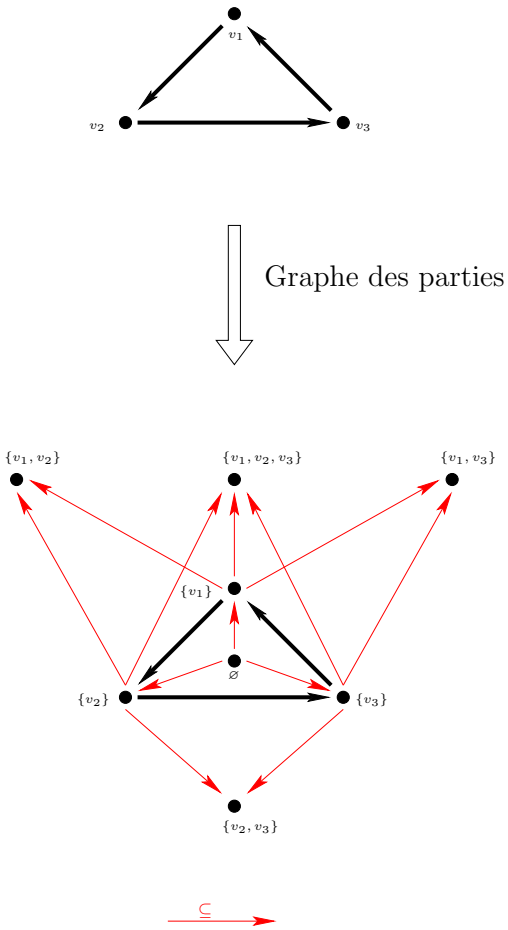


FIGURE 3.13 – Graphe des parties (finies) d'un graphe.

Définition 3.3.16 (Interprétation à ensembles finis)**.** Un graphe H est interprétation à ensemble finis d'un graphe G si H est FO-interprétation du graphe des parties finies de G.

Observation 3.3.17. Soit G un graphe dont la WMSO-théorie est décidable et H une interprétation à ensembles finis de G. Alors la FO-théorie de H est décidable.

3.3.4 Hiérarchies de graphes

Hiérarchie à pile

Les transformations de graphes présentées dans la partie précédente permettent de définir la hiérarchie à pile, une hiérarchie de graphes dont la théorie du second ordre monadique est décidable. Les graphes considérés ici, le sont à isomorphisme près : nous ne distinguons pas deux graphes isomorphes.

Définition 3.3.18. La hiérarchie à pile $\mathcal{C}_0 \subsetneq \mathcal{C}_1 \dots$ est définie comme suit. Le premier niveau $\mathcal{C}_0$ est constitué des graphes finis. Chaque niveau supérieur $\mathcal{C}_{n+1}$ est constitué des substitutions rationnelles inverses des dépliés des graphes du niveau précédent $\mathcal{C}_n$.

Comme le dépliage depuis un sommet MSO-définissable et la MSO-interprétation préservent la décidabilité de la logique du second ordre monadique, on en déduit la proposition suivante.

Proposition 3.3.19. *Tout graphe appartenant à la hiérarchie à pile a une* MSO-*théorie décidable.*

La robustesse de la hiérarchie à pile est notamment illustrée par le fait que des transformations, a priori plus générales que celles utilisées dans la définition précédente, ne permettent pas d'obtenir plus de graphes, à isomorphisme près.

Proposition 3.3.20 ([5])**.** *Soit $n \geqslant 1$. Les graphes du niveau n de la hiérarchie à pile sont les MSO-interprétations des dépliés du niveau précédent.*

Il se trouve que chaque niveau de la hiérarchie à pile peut également être décrit à partir d'un générateur, qui ne sera autre que l'arbre binaire infini pour le niveau 1 et les structures arborescentes itérées de celui-ci pour les niveaux supérieurs. Notons T_2 l'arbre binaire infini défini par

$$T_2 := \{u \xrightarrow{A} uA \mid u \in \{A, B\}^*\} \ \cup \ \{u \xrightarrow{B} uB \mid u \in \{A, B\}^*\}$$

Soit $(\sharp_i)_{i\in\mathbb{N}} \in (\Lambda \setminus \{A, B\})^{\mathbb{N}}$ une suite d'étiquettes distinctes de A et B et distinctes deux à deux. On définit pour tout $n \geqslant 0$, la structure arborescente itérée n fois de l'arbre binaire infini, notée $\text{Treegraph}^n(T_2)$, par : $\text{Treegraph}^0(T_2) := T_2$, et pour tout $n \geqslant 1$,

$$\text{Treegraph}^n(T_2) := \text{Treegraph}_{\sharp_n}(\text{Treegraph}^{n-1}(T_2))$$

Voir la définition 3.3.4.

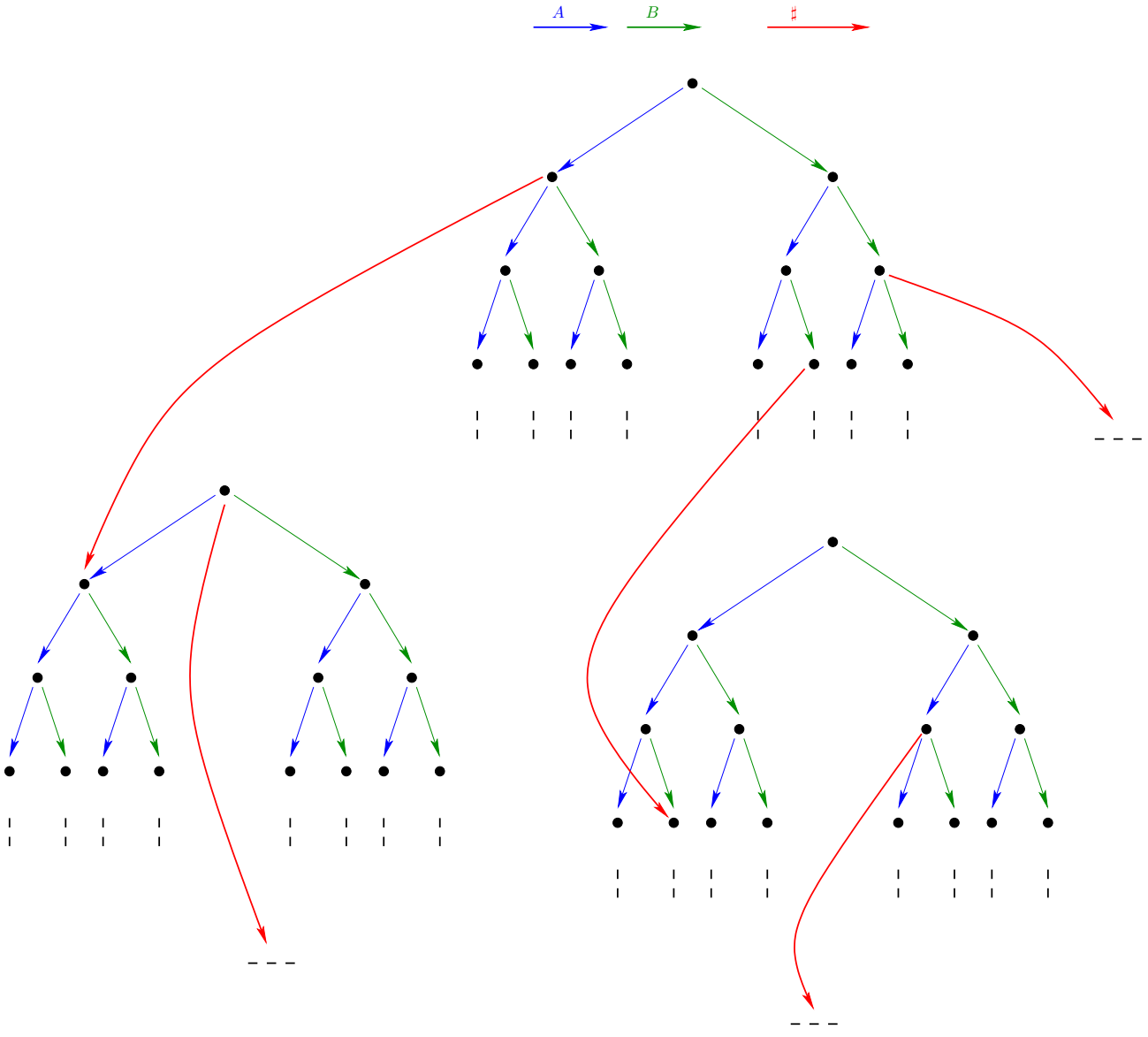


FIGURE 3.14 – La structure arborescente Treegraph(T_2) de l'arbre binaire infini.

Exemple 3.3.21. Voir la figure 3.14 pour la structure arborescente de l'arbre binaire infini. Celui-ci est générateur du niveau 2 de la hiérarchie à pile.

Une première façon d'obtenir le niveau n de la hiérarchie à pile à partir du générateur $\text{Treegraph}^{n-1}(T_2)$ est de procéder par transduction monadique.

Proposition 3.3.22. *Soit $n \geqslant 1$. Les graphes du niveau n de la hiérarchie à pile sont les transductions monadiques de* $\text{Treegraph}^{n-1}(T_2)$.

Observons également que chaque niveau de la hiérarchie à pile est fermé par MSO-interprétation.

Exemple 3.3.23. Soit $T_1 = \{u \xrightarrow{a} ua \mid u \in \{a\}^*\}$ la demi-droite. Sa structure arborescente $\text{Treegraph}_\sharp(T_1)$ appartient au niveau 2 de la hiérarchie à pile. Voir la figure 3.16 pour un automate dont le graphe sous-jacent est MSO-interprétation de $\text{Treegraph}_\sharp(T_1)$ (voir la figure 3.15).

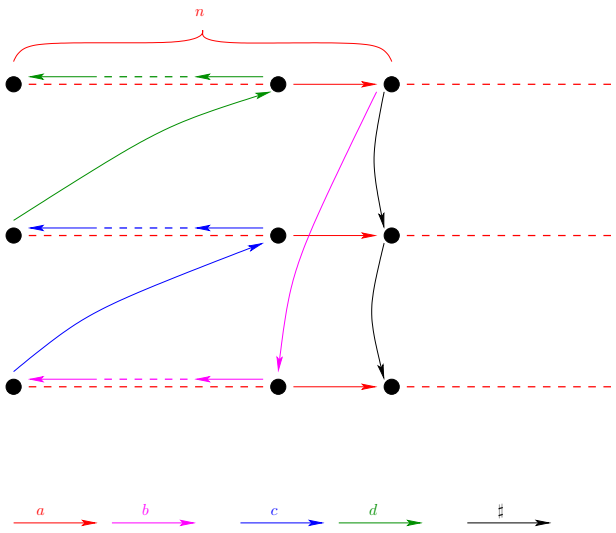


FIGURE 3.15 – Une MSO-interprétation de $\text{Treegraph}_\sharp(T_1)$, la structure arborescente de la demi-droite (voir l'exemple 3.3.23).

On peut aussi procéder par substitutions rationnelles inverses des marquages rationnels du générateur.

Un *marquage rationnel* d'un Σ_1-graphe G par un symbole $\sharp \notin \Sigma_1$ selon un langage rationnel $L \subseteq (\Sigma_1 \cup \overline{\Sigma_1})^*$ et depuis un sommet $r \in V_G$ est le graphe $\sharp_L(G, r)$ défini par

$$\sharp_L(G,r) := \{(v,0) \xrightarrow{a} (w,0) \mid v \xrightarrow[G]{a} w\} \cup \{(v,0) \xrightarrow{\sharp} (v,1) \mid r \xrightarrow[\overline{G}]{L} v\}$$

Proposition 3.3.24. *Soit $n \geqslant 1$. Les graphes du niveau n de la hiérarchie à pile sont les substitutions rationnelles inverses des marquages rationnels de* $\text{Treegraph}^{n-1}(T_2)$ *depuis sa racine.*

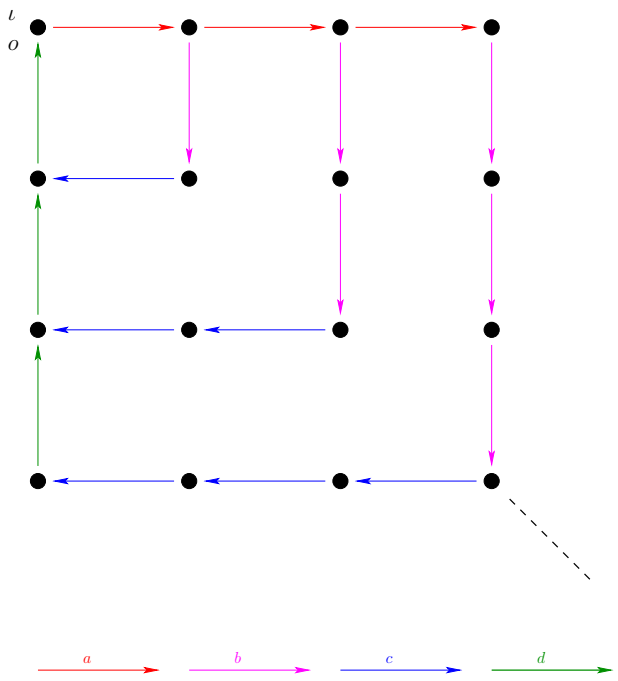


FIGURE 3.16 – Un automate dont le langage accepté est $\{a^n b^n c^n d^n \mid n > 0\}$ (voir l'exemple 3.3.23).

Hiérarchie arbre-automatique

La hiérarchie arbre-automatique [14] est une hiérarchie de graphes dont la théorie du premier ordre est décidable : chaque niveau est constitué des interprétations à ensembles finis des arbres du niveau correspondant de la hiérarchie à pile.

3.3.5 Substitutions inverses de l'arbre binaire infini

Dans cette partie, on considère l'arbre binaire avec arcs retours

$$\tilde{T}_2 := \{u \xrightarrow{x} ux \mid u \in \{a,b\}^*, x \in \{a,b\}\} \cup \{ux \xrightarrow{\overline{x}} u \mid u \in \{a,b\}^*, x \in \{a,b\}\}$$

ainsi que la classe de ses marquages rationnels depuis la racine, c'est-à-dire la classe des graphes de la forme $\sharp_L(\tilde{T}_2, \varepsilon)$, où $L \in \mathrm{Rat}(\{a,b,\overline{a},\overline{b}\}^*)$

On décrit diverses classes de graphes rencontrés précédemment (et appartenant au premier niveau arbre-automatique) de la façon uniforme suivante : étant donnée une famille F de langages, où $F \subseteq \mathcal{P}(\{a,b,\overline{a},\overline{b}\}^*)$, et notant F_Σ l'ensemble des substitutions h de la forme $h : \Sigma^* \longrightarrow F$, on considère la classe $\mathfrak{G}_F$ des Σ-graphes obtenus par application inverse des substitutions appartenant à F_Σ aux marquages rationnels depuis la racine de $\tilde{T}_2$

$$\mathfrak{G}_F := \{h^{-1}(\sharp_L(\tilde{T}_2, \varepsilon)) \mid h \in F_\Sigma, L \in \mathrm{Rat}\,\Sigma^*\}$$

Il se trouve que les graphes obtenus par substitution finie inverse de l'arbre binaire avec arcs retours sont exactement les graphes suffixes de réécriture de mots. Notons Fin la classe des langages finis de $\{a, b, \overline{a}, \overline{b}\}$-mots.

Théorème 3.3.25. *Les assertions suivantes sont équivalentes :*

1. $G \in \mathfrak{G}_{\mathrm{Fin}}$
2. *G est isomorphe à un Σ-graphe de réécriture suffixe de mots.*

Considérons à présent la famille $\mathrm{Rat}(\{a, b, \overline{a}, \overline{b}\}^*)$ des langages rationnels de $\{a, b, \overline{a}, \overline{b}\}$-mots.

Commençons par définir la notion de *graphe d'un système reconnaissable de réécriture de mots*. Un tel graphe est une union finie de graphes de la forme :

$$W(U \xrightarrow{a} V)$$

où W, U et V sont des langages rationnels de mots, a est une étiquette et

$$W(U \xrightarrow{a} V) = \{wu \xrightarrow{a} wv \mid w \in W, u \in U, v \in V\}$$

Exemple 3.3.26. Un exemple de graphe d'un système reconnaissable de réécriture de mots est donné par la demi-droite des entiers naturels, munie de son ordre usuel. Voir la figure 3.17 et considérer le système reconnaissable de réécriture de mots suivant :

$$x^*(\varepsilon \xrightarrow{S_0} x) \ \cup \ x^*(\varepsilon \xrightarrow{\leqslant^{\mathbb{N}}} x^*)$$

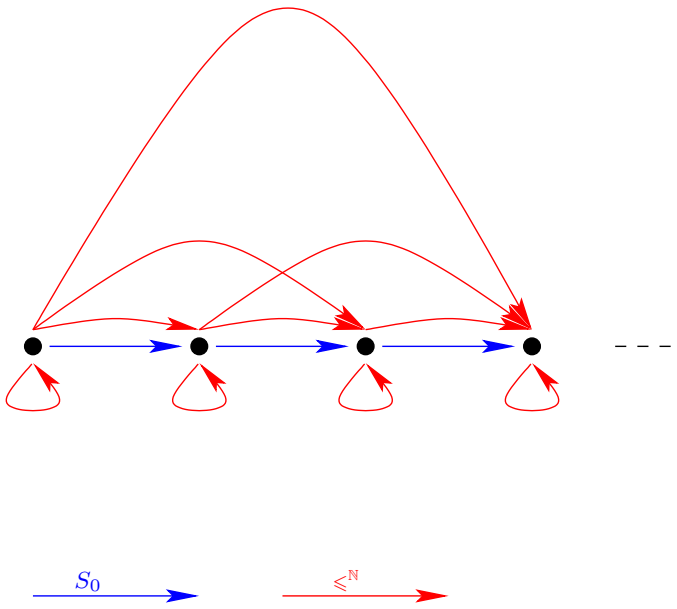


FIGURE 3.17 – Demi-droite des entiers naturels munie de son ordre usuel.

Théorème 3.3.27. *Les assertions suivantes sont équivalentes :*

1. $G \in \mathfrak{G}_{\mathrm{Rat}(\{a,b,\overline{a},\overline{b}\}^*)}$
2. *G est isomorphe à un Σ-graphe d'un sytème reconnaissable de réécriture de mots.*
3. *G appartient au niveau 1 de la hiérarchie à pile.*

Graphes rationnels

Un *langage linéaire* est le langage d'une grammaire dont les productions sont de la forme :

$$X \ \to \ \varepsilon \text{ ou } X \ \to \ uYv$$

où u, v sont des mots terminaux et X et Y sont des lettres non-terminales. En particulier, un langage linéaire est un langage algébrique.

Notons Lin la famille des $\{a, b, \overline{a}, \overline{b}\}$-langages linéaires et $\overline{\mathrm{Lin}} \subseteq \mathrm{Lin}$, la sous-famille engendrée par des grammaires algébriques dont le membre droit de toute production est ε ou bien uXv, où $u \in \{\overline{a}, \overline{b}\}^*$, $v \in \{a, b\}^*$ et X est une lettre non-terminale.

Théorème 3.3.28 ([31])**.** *Les assertions suivantes sont équivalentes :*

1. $G \in \mathfrak{G}_{\overline{\mathrm{Lin}}}$
2. *G est isomorphe à un Σ-graphe rationnel.*

A présent, considérons la question de la décidabilité du problème de satisfaction de la logique du premier ordre pour les graphes rationnels.

Proposition 3.3.29. *Le problème de la satisfaction pour un graphe rationnel donné d'une formule du premier ordre donnée est indécidable.*

En effet, on peut y réduire le problème (indécidable) de correspondance de Post (P.C.P.). Rappelons qu'une instance de ce problème est la donnée d'une suite de couples de mots (sur un alphabet contenant au moins deux lettres) (u_i, v_i) $(i \in \{0, \ldots, n\})$ et que la question posée est : existe-t-il une suite $0 \leqslant i_1, i_2, \ldots, i_m \leqslant n$ telle que

$$u_0 u_{i_1} \ldots u_{i_m} = v_0 v_{i_1} \ldots v_{i_m} \, ?$$

Associant à une instance de P.C.P. le transducteur décrit dans l'exemple 3.2.7, on constate que résoudre P.C.P. revient à déterminer si le graphe rationnel défini par ce transducteur satisfait l'énoncé du premier ordre : $\exists s \ s \xrightarrow{a} s$ (où a est l'étiquette de la relation définie par le transducteur).

Cependant, si l'on se restreint aux arbres rationnels, on obtient un résultat positif de décidabilité de la logique du premier ordre.

Proposition 3.3.30 ([4])**.** *La théorie du premier ordre d'un arbre rationnel est décidable.*

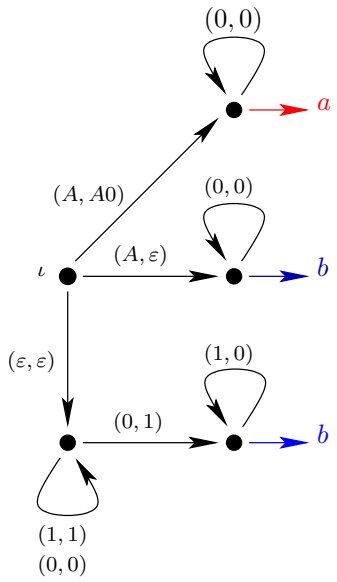


FIGURE 3.18 – Un transducteur pour un arbre rationnel sur l'alphabet $\{A, 0, 1\}$.

Exemple 3.3.31. Voir les figures 3.18 et 3.19 pour un exemple de transducteur définissant un arbre rationnel.

Le résultat précédent est difficilement généralisable tant d'un point de vue de l'expressivité de la logique que d'un point de vue structurel, comme le montrent les deux propositions suivantes (voir le paragraphe 2.4.2 pour un rappel concernant la logique du premier ordre avec accessibilité rationnelle).

Proposition 3.3.32 ([4])**.** *Il existe un arbre rationnel dont la théorie du premier ordre avec accessibilité rationnelle n'est pas décidable.*

Proposition 3.3.33 ([4, 46])**.** *Il existe un DAG rationnel dont la théorie du premier ordre n'est pas décidable.*

Dans le chapitre 5, nous mettrons notamment en évidence une sous-classe de *DAG* rationnels dont la théorie du premier ordre avec accessibilité est décidable. Voir la figure 3.20.

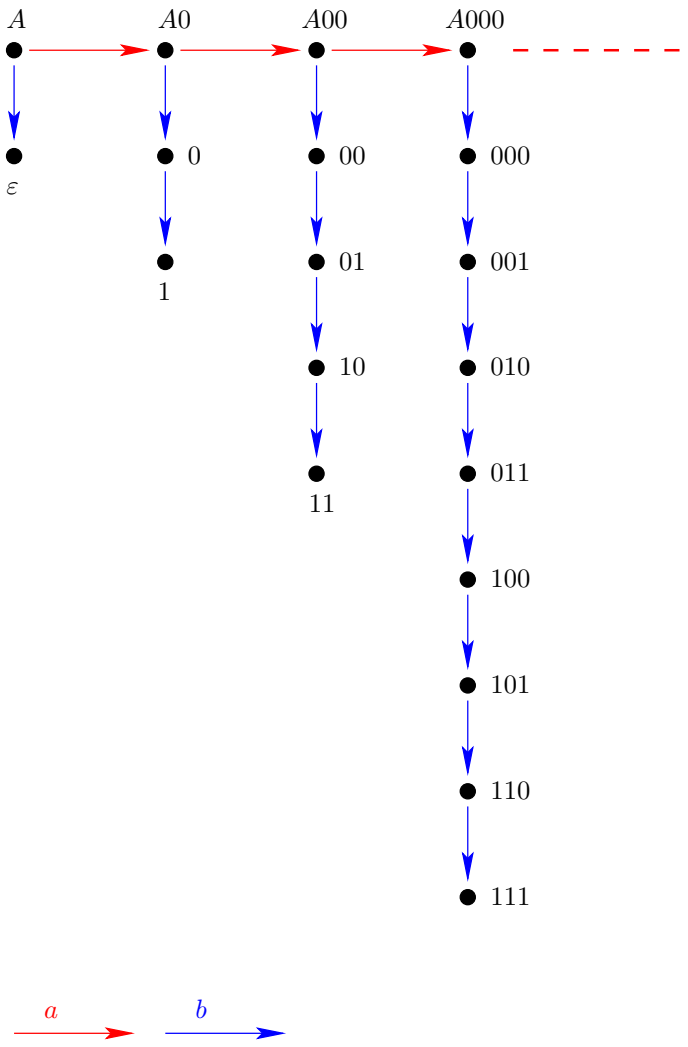


FIGURE 3.19 – Un arbre rationnel.

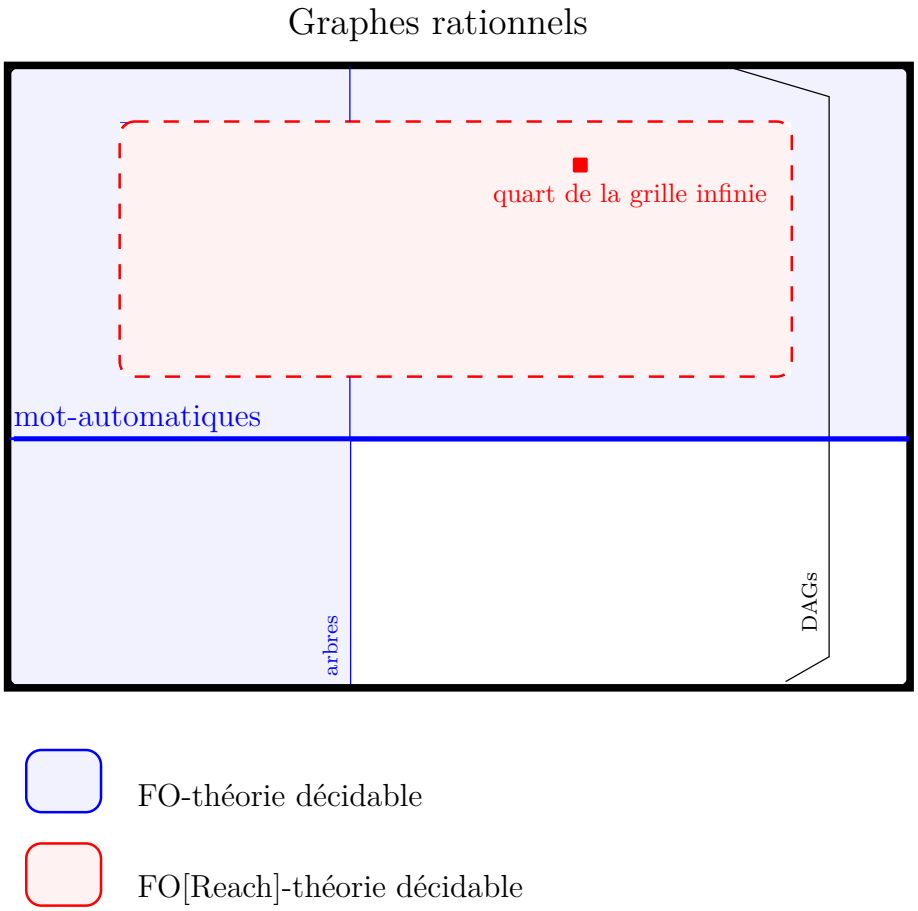


FIGURE 3.20 – Vers une classe de DAG mot-automatiques (voir la définition 5.2.6) dont la FO[Reach]-théorie est décidable.

Chapitre 4

Traces et régularité par niveaux

Dans ce chapitre, nous commençons par rappeler quelques généralités à propos des traces de Mazurkiewicz. Ensuite, nous considérons la notion de langage de traces régulier par niveaux : un langage de traces est régulier par niveaux si l'ensemble des formes normales de Foata de ses éléments constitue un langage régulier de mots. On observera, en particulier, qu'un langage de traces reconnaissable (voir paragraphe 2.2.6) est régulier par niveaux mais que la réciproque n'est pas vraie.

4.1 Généralités

Soit Σ un alphabet.

Une *relation de dépendance* D sur Σ est une relation binaire réflexive et symétrique. La paire (Σ, D) est appelée *alphabet de dépendance*. Le complémentaire de D est la *relation d'indépendance* $I := \Sigma^2 \backslash D$. Notons $\equiv_D$ la plus petite des congruences $\equiv$ de Σ^* telle que

$$(a, b) \in I \implies ab \equiv ba$$

La (Σ, D)-trace d'un mot $w \in \Sigma^*$ est sa classe d'équivalence pour la relation d'équivalence $\equiv_D$. Elle est notée $[w]$.

Observons que deux Σ-mots $\equiv_D$-équivalents ont la même longueur. La longueur d'une trace t est la longueur de n'importe quel mot lui appartenant et est notée $|t|$.

Le monoïde quotient $\Sigma^* / \equiv_D$ est appelé *monoïde de traces* de l'alphabet de dépendance (Σ, D) et est noté $M(\Sigma, D)$. Ainsi le produit du monoïde de traces est défini pour tout $[u], [v] \in M(\Sigma, D)$ par $[u][v] = [uv]$.

Le graphe de Cayley de $M(\Sigma, D)$ est

$$\text{Cayley}_{M(\Sigma,D)} := \{t \xrightarrow{a} t[a] \mid t \in M(\Sigma, D), a \in \Sigma\}$$

Exemple 4.1.1. Supposons que $\Sigma = \{a, b, c\}$ et $I = \{(a, c), (c, a), (b, c), (c, b)\}$. Voir la figure 4.1 pour $\text{Cayley}_{M(\Sigma,D)}$.

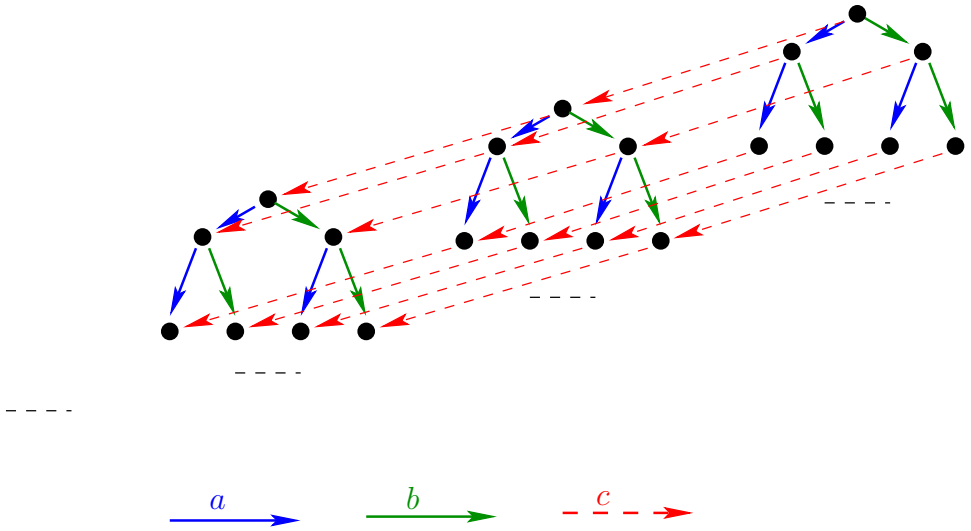


FIGURE 4.1 – Le graphe de Cayley d'un monoïde de traces (voir l'exemple 4.1.1).

La relation binaire de préfixité $\sqsubseteq$ sur $M(\Sigma, D)$, définie par $t \sqsubseteq t'$ si et seulement s'il existe $s \in M(\Sigma, D)$ telle que $ts = t'$, est un ordre partiel.

Notons $\mathrm{Total}_\Sigma := \Sigma \times \Sigma$ la relation de dépendance totale sur Σ et $\mathrm{Id}_\Sigma := \{(a, a) \mid a \in \Sigma\}$ la relation d'égalité. Remarquons que dans le cas où $D = \mathrm{Total}_\Sigma$, le monoïde de traces $M(\Sigma, D)$ s'identifie au monoïde libre Σ^*.

Forme normale de Foata

Considérons l'alphabet fini $I_D := \{A \subseteq \Sigma \mid \forall a_1 \neq a_2 \in A\ (a_1, a_2) \in I\}$ des parties indépendantes de Σ, et notons $\Pi_{I_D} : I_D^* \to M(\Sigma, D)$ le morphisme canonique défini par

$$\Pi_{I_D}(\varnothing) = [\varepsilon] \text{ et } \Pi_{I_D}(\{a_1, \cdots, a_n\}) = [a_1 \dots a_n] \quad (n \geqslant 1)$$

Etant donné $P \subseteq I_D$, notons Π_P la restriction de Π_{I_D} à P^*. Un P-mot U code la trace $\Pi_P(U)$.

Considérons la relation binaire $\rhd$ sur $I_D^- := I_D \setminus \{\varnothing\}$ définie par :

$$A \rhd B \iff \forall b \in B\ \exists a \in A\ aDb$$

Notons $\mathbf{F} \subseteq I_D^{-*}$ l'ensemble des mots sur I_D^{-*} qui sont des $\rhd$-chemins :

$$\mathbf{F} := \{A_1 \dots A_n \mid A_1 \rhd \cdots \rhd A_n, n \geqslant 0\}$$

Le morphisme surjectif Π_{I_D} n'est pas injectif. En effet, supposons par exemple que $\Sigma = \{a, b\}$ et $I = \{(a, b), (b, a)\}$, alors $\Pi_{I_D}(\{a, b\}) = \Pi_{I_D}(\{a\}\{b\})$.

La proposition suivante montre que toute trace peut se coder par un unique I_D^--mot appartenant à $\mathbf{F}$.

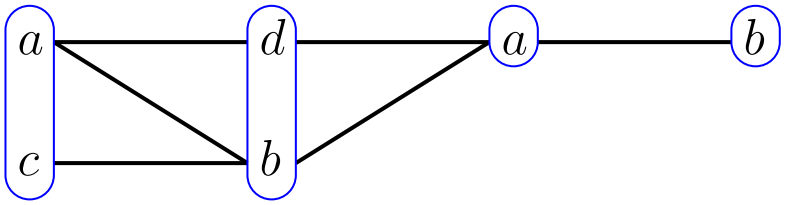


FIGURE 4.2 – La forme normale de Foata de $[acbdab]$ (voir l'exemple 4.1.3).

Proposition 4.1.2 (Forme normale de Foata)**.** *Soit $t \in M(\Sigma, D)$. Il existe un unique I_D^--mot $\lceil t \rceil_{\mathbf{F}} = A_1 \cdots A_p \in \mathbf{F}$ $(p \geqslant 0)$, la forme normale de Foata de t, tel que $\Pi_{I_D}(A_1 \cdots A_p) = t$.*

Exemple 4.1.3. Supposons que $\Sigma = \{a, b, c, d\}$ et

$$D = \mathrm{Id}_\Sigma \cup \{(a,b),(b,a),(a,d),(d,a),(b,c),(c,b)\}$$

En particulier, aIc, bId, cId. La forme normale de Foata de $t = [acbdab]$ est

$$\lceil t \rceil_{\mathbf{F}} = \{a,c\}\{b,d\}\{a\}\{b\}$$

(voir la figure 4.2).

Dans la suite et étant donnée une trace t, notons $\|t\|$ la longueur de sa forme normale de Foata.

L'ensemble des formes normales de Foata $\mathbf{F}$ est accepté par un automate fini.

Lemme 4.1.4. *L'ensemble $\mathbf{F}$ des formes normales de Foata est un langage régulier de I_D^{-*}-mots.*

Démonstration. L'ensemble $\mathbf{F}$ est accepté par l'I_D^--automate fini $\mathcal{A}_{\mathbf{F}}$ suivant :

- $\bot \xrightarrow{A} A : A \in I_D^-$
- $A \xrightarrow{B} B : A \rhd B$
- le sommet initial est $\bot \notin I_D^-$
- tous les sommets sont finaux (même $\bot$).

□

Reconnaissabilité

Un (Σ, D)-*language de traces* est une partie de $M(\Sigma, D)$. Si $\mathcal{L}$ est un langage de traces, alors $\bigcup \mathcal{L} = \{w \in \Sigma^* \mid [w] \in \mathcal{L}\}$ est un langage de Σ-mots. Si L est un langage de mots, alors $[L]$ est le langage de traces défini par $[L] := \{[w] \in M(\Sigma, D) \mid w \in L\}$. En particulier, $[\bigcup \mathcal{L}] = \mathcal{L}$ et $\bigcup [L] \supseteq L$. Précisément, $\bigcup [L] = \{w \mid \exists w' \in L \; w \equiv_D w'\}$.

Les notions de résiduation et de produit de langages ont été définies pour un monoïde quelconque au paragraphe 2.2.6.

Exemple 4.1.5. Supposons que $\Sigma = \{a, b\}$ et $D = \mathrm{Id}_\Sigma$. En particulier aIb. Le résidu à gauche par $[ab]$ de $\mathcal{L} = \{[ab], [abaa], [aaa], [aabbb]\}$ est :

$$[ab]^{-1}\mathcal{L} = \{\varepsilon, [aa], [abb]\}$$

Le produit à droite de $\mathcal{L}$ par $[a]$ est :

$$\mathcal{L}[a] = \{[aab], [aaaab], [aaaa], [aaabbb]\}$$

La classe des langages de traces reconnaissables est notée $\mathrm{Rec}(M(\Sigma, D))$.

Proposition 4.1.6 ([18])**.** $\mathrm{Rec}(M(\Sigma, D))$ *est une algèbre Boole fermée par concaténation.*

Remarque 4.1.7. Si $D = \Sigma^2$, alors $\mathrm{Rec}(M(\Sigma, D)) = \mathrm{Reg}(\Sigma^*)$.

Rappelons diverses caractérisations des langages reconnaissables de traces (voir le paragraphe 2.2.6).

Proposition 4.1.8. *Les assertions suivantes sont équivalentes :*

1. $\mathcal{L}$ *est reconnaissable,*
2. *l'ensemble* $\{t^{-1}\mathcal{L} \mid t \in M(\Sigma, D)\}$ *des résidus à gauche de* $\mathcal{L}$ *est fini,*
3. *l'ensemble* $\{\mathcal{L}t^{-1} \mid t \in M(\Sigma, D)\}$ *des résidus à droite de* $\mathcal{L}$ *est fini.*

Considérons un alphabet fini P et $\pi : P^* \to M(\Sigma, D)$ un morphisme surjectif. Par exemple, P pourrait être Σ, I_D^- ou I_D. Pour une trace t, si nous voyons $\pi^{-1}(t)$ comme l'ensemble de ses P-codages, la proposition suivante montre que la reconnaissabilité d'un langage de traces est équivalente à la régularité de l'ensemble des P-codages de ses éléments.

Proposition 4.1.9. $\mathcal{L} \in \mathrm{Rec}(M(\Sigma, D))$ *si et seulement si* $\pi^{-1}(\mathcal{L})$ *est régulier.*

Corollaire 4.1.10. $\mathcal{L} \in \mathrm{Rec}(M(\Sigma, D))$ *si et seulement si* $\bigcup \mathcal{L}$ *est un langage régulier.*

4.2 Régularité par niveaux

Le langage de traces $[(ab)^*]$, où aIb, n'est pas reconnaissable puisque son union, l'ensemble des mots sur $\{a, b\}$ avec le même nombre d'occurrences de a et b, n'est pas régulier. Néanmoins, l'ensemble $\{a, b\}^*$ des formes normales de Foata de ses éléments est régulier. Cela suggère de considérer une notion plus faible de reconnaissabilité.

Définition 4.2.1. $\mathcal{L} \subseteq M(\Sigma, D)$ est régulier par niveaux si le langage (de mots) $\lceil \mathcal{L} \rceil_{\mathbf{F}}$ est régulier.

Puisque $\lceil \mathcal{L} \rceil_{\mathbf{F}} = \Pi_{I_D}^{-1}(\mathcal{L}) \cap \mathbf{F}$, tout langage de traces reconnaissable est régulier par niveaux. Notons également que tout langage de traces régulier par niveaux est une partie rationnelle du monoïde $M(\Sigma, D)$. Notons LevelReg$(M(\Sigma, D))$ la classe des langages de traces réguliers par niveaux du monoïde de traces $M(\Sigma, D)$.

Proposition 4.2.2 ([27, 2])**.** LevelReg$(M(\Sigma, D))$ *est une algèbre de Boole.*

Démonstration. Observons qu'étant donnés deux langages de traces réguliers par niveaux $\mathcal{L}_1$ et $\mathcal{L}_2$, l'ensemble des formes normales de Foata de leur intersection (respectivement l'ensemble des formes normales de Foata du complémentaire de $\mathcal{L}_1$) est $\lceil \mathcal{L}_1 \cap \mathcal{L}_2 \rceil_{\mathbf{F}} = \lceil \mathcal{L}_1 \rceil_{\mathbf{F}} \cap \lceil \mathcal{L}_2 \rceil_{\mathbf{F}}$ (respectivement $\left\lceil \overline{\mathcal{L}_1} \right\rceil_{\mathbf{F}} = \overline{\lceil \mathcal{L}_1 \rceil_{\mathbf{F}}}$). La proposition est donc conséquence du fait que la classe des langages réguliers forme une algèbre de Boole. □

La classe LevelReg$(M(\Sigma, D))$ n'est cependant pas close par concaténation [27].

Exemple 4.2.3. Considérons la concaténation des deux langages de traces $[(ab)^*]$ et $[(bc)^*]$, avec $D = \{(a,a),(b,b),(c,c)\}$. L'ensemble des formes normales de Foata de ses éléments

$$\begin{array}{rl} & \Pi_{\mathbf{F}}^{-1}([(ab)^*] \cdot [(bc)^*]) \\ = & \{\{a,b,c\}^k \{b,c\}^* \{b\}^k \mid k \geqslant 0\} \cup \{\{a,b,c\}^k \{a,b\}^* \{b\}^k \mid k \geqslant 0\} \end{array}$$

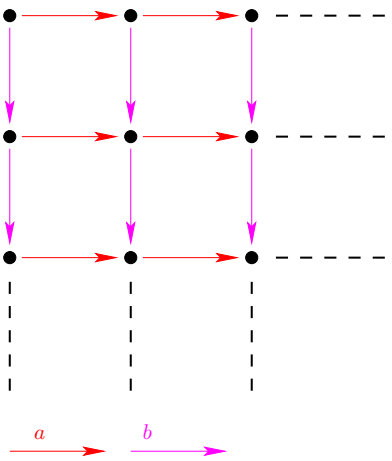
n'est pas régulier.

Chapitre 5

Dépliage concurrent d'un graphe fini concurrent

5.1 Introduction

Comme on peut y réduire le problème de l'arrêt des machines de Turing, la MSO-théorie du quart de la grille infinie n'est pas décidable. Le quart de la grille infinie n'appartient donc pas à la hiérarchie à pile et n'est pas, en particulier, le déplié d'un graphe fini.



Néanmoins, comme il s'agit d'un graphe de réécriture suffixe de termes clos (graphe GTR), sa FO[Reach]-théorie est décidable (paragraphe 5.4). En effet, un graphe de réécriture suffixe de termes clos (graphe GTR), muni de la relation d'accessibilité est arbre-automatique et sa FO[Reach]-théorie est donc décidable [17]. En fait, même la théorie du premier ordre étendue par l'opérateur de fermeture transitive pour les relations définissables au premier ordre du quart de la grille infinie reste décidable [49]. Mais considérons maintenant l'arbre du quart de la grille infinie : tout sommet du quart d'une grille infinie est source d'un arc étiqueté par un nouveau

symbole et dont le but est l'origine d'un nouveau quart de grille infinie et ainsi de suite (paragraphe 5.4). Nous montrerons que sa FO[Reach]-théorie est décidable et que néanmoins il n'est pas un graphe GTR.

En fait, nous nous intéressons, plus généralement, à une classe de graphes qui modélisent les exécutions de systèmes concurrents. Pour ce faire, nous considérons les traces de Mazurkiewicz.

Soit un système reconnaissable de réécriture de traces, c'est-à-dire un ensemble fini de règles de la forme $\mathcal{U} \cdot (\mathcal{V} \xrightarrow{\lambda} \mathcal{W})$ où $\mathcal{U}$, $\mathcal{V}$, $\mathcal{W}$ sont des langages de traces reconnaissables sur un monoïde de traces $M(\Sigma, D)$, λ une étiquette. Considérons son graphe de réécriture : l'ensemble des arcs de la forme $ts \xrightarrow{\lambda} ts'$ tel qu'il existe une règle de réécriture $\mathcal{U} \cdot (\mathcal{V} \xrightarrow{\lambda} \mathcal{W})$ avec $t \in \mathcal{U}$, $s \in \mathcal{V}$, $s' \in \mathcal{W}$. Si D est la relation totale, alors un tel graphe est au premier niveau de la hiérarchie à pile car il s'agit d'un graphe de réécriture d'un système reconnaissable de réécriture de mots. Si D est l'égalité, alors il s'agit d'une notion plus générale de réseau de Petri, dont on discutera dans le chapitre suivant. Dans tous les cas, nous montrerons qu'un automate dont le graphe sous-jacent est graphe de réécriture d'un système reconnaissable de traces et dont les ensembles de sommets initiaux et finaux sont réguliers par niveaux (automate RTL), est mot-automatique [1], même si les contextes des règles de réécriture ne sont supposés que réguliers par niveaux.

La théorie du premier ordre d'un automate RTL est donc décidable. Nous montrerons également que le problème de la vacuité du langage accepté par un automate RTL n'est pas décidable car l'on peut y réduire le problème (indécidable) de l'arrêt des machines de Minsky à deux compteurs.

Nous montrerons que la FO[Reach]-théorie du déplié concurrent d'un graphe fini concurrent à partir de tout sommet est décidable, en montrant qu'un tel déplié enrichi de la relation d'accessibilité, est un graphe RTL. Cela étend un résultat de Madhusudan [25] sur la décidabilité de la théorie du premier ordre des structures d'événements trace-régulières [45]. Nous observerons que le quart de la grille infinie ainsi que son arbre sont des dépliés concurrents de graphes finis concurrents. Nous en déduirons, plus généralement, que la FO[Reach]-théorie de tout automate dont le graphe sous-jacent est un déplié concurrent d'un graphe fini concurrent et dont les ensembles de sommets initiaux et finaux sont réguliers par niveaux, est décidable.

Nous définirons l'arbre d'un graphe et nous montrerons que si l'arbre d'un graphe est graphe de réécriture suffixe d'un système de réécriture de termes clos, alors il est finiment décomposable par taille. Si un graphe est finiment décomposable par taille, alors il appartient au premier niveau de la hiérarchie à pile. Par conséquent, la classe des dépliés concurrents des graphes finis concurrents contient des graphes non GTR mais dont la FO[Reach]-théorie est décidable. L'arbre du quart de la grille infinie est un exemple de tel graphe.

Les résultats de ce chapitre ont été publiés dans [27].

1. Un graphe est mot-automatique si chacune de ses relations est reconnaissable par un transducteur synchronisé fini.

5.2 Système de réécriture de traces

Les graphes au premier niveau de la hiérarchie à pile sont les graphes de réécriture des systèmes reconnaissables de réécriture de mots. Un tel système est un ensemble fini de règles (de réécriture) de la forme $U \cdot (V \to W)$, où U (le contexte), V et W sont des langages réguliers. Dans la suite, nous considérons des systèmes reconnaissables de réécriture de traces, à contextes réguliers par niveaux (RTL) et membres gauches et droits reconnaissables. Nous prouvons qu'un automate, dont le graphe sous-jacent est graphe de réécriture d'un tel système et dont les ensembles de sommets initiaux et finaux sont réguliers par niveaux, est mot-automatique, en codant l'ensemble de ses sommets par leurs formes normales de Foata.

Définition 5.2.1. Un système reconnaissable $\mathbf{R}$ de réécriture de traces à contextes réguliers par niveaux (RTL) sur $M(\Sigma, D)$ est la donnée d'un ensemble fini de règles (de réécriture) de la forme

$$\mathcal{U} \cdot (\mathcal{V} \xrightarrow{\lambda} \mathcal{W})$$

où $\mathcal{U} \in \mathrm{LevelReg}(M(\Sigma, D))$, $\mathcal{V}, \mathcal{W} \in \mathrm{Rec}(M(\Sigma, D))$ et $\lambda \in \Lambda$.

Le graphe de réécriture $\mathrm{Gr}_{\mathbf{R}}$ du RTL $\mathbf{R}$ est le Λ-graphe sur $M(\Sigma, D)$ défini par

$$\mathrm{Gr}_{\mathbf{R}} := \{[uv] \xrightarrow{\lambda} [uw] \mid \exists\ \mathcal{U} \cdot (\mathcal{V} \xrightarrow{\lambda} \mathcal{W}) \in \mathbf{R}, [u] \in \mathcal{U}, [v] \in \mathcal{V}, [w] \in \mathcal{W}\}$$

Remarque 5.2.2. Observons qu'une règle de réécriture de la forme $\mathcal{U} \cdot ([\varepsilon] \xrightarrow{\lambda} [\varepsilon])$ conduit dans le graphe de réécriture $\mathrm{Gr}_{\mathbf{R}}$ à des boucles étiquetées par λ sur toutes les traces appartenant à $\mathcal{U}$.

Un *automate de système reconnaissable de réécriture de traces à contextes réguliers par niveaux* (automate RTL) sur $M(\Sigma, D)$ est un automate de la forme

$$G \ \cup\ \{\iota\} \times I \ \cup\ \{o\} \times F$$

où G est le graphe de réécriture d'un système RTL sur $M(\Sigma, D)$ et I (respectivement F) est un langage régulier par niveaux sur $M(\Sigma, D)$ de traces initiales (respectivement finales).

Exemple 5.2.3. Supposons que $D = \{(a, a), (b, b)\}$ et considérons le RTL suivant :
$[(a + b)^*] \cdot ([\varepsilon] \xrightarrow{a} [a])$
$[(a + b)^*] \cdot ([\varepsilon] \xrightarrow{b} [b])$
$[(ab)^*] \cdot ([\varepsilon] \xrightarrow{f} [\varepsilon])$
Son graphe de réécriture est le quart de la grille infinie avec une boucle étiquetée par le symbole f sur chaque sommet de sa diagonale (voir la figure 5.1). Sa théorie du second ordre monadique n'est pas décidable car le quart de la grille infinie y est MSO-interprétable. En particulier, il n'appartient pas à la hiérarchie à pile.

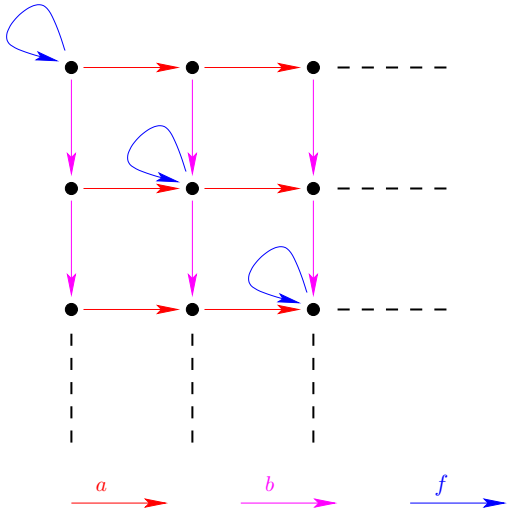


FIGURE 5.1 – La diagonale du quart de grille infinie (exemple 5.2.3)

Exemple 5.2.4. Supposons que $D = \{(a,a),(b,b),(c,c)\}$ et considérons le RTL suivant :

$[(abc)^*] \cdot ([\varepsilon] \xrightarrow{a} [abc])$

$[(abc)^*(ac)^*] \cdot ([b] \xrightarrow{b} [\varepsilon])$

$[(abc)^*(ac)^*] \cdot ([ac] \xrightarrow{c} [\varepsilon])$

Son graphe de réécriture (voir la figure 5.2) n'appartient pas à la hiérarchie à pile car sa théorie du second ordre monadique n'est pas décidable. En effet, le quart de la grille infinie y est MSO-interprétable ; considérer la MSO-interprétation suivante.

— $\delta(x) = \quad (x = x)$

— $\phi_a(x,y) = \quad (x \xrightarrow{a} y \ \vee \ \exists z \ (z \xrightarrow{b} x \ \wedge \ y \xrightarrow{c} z))$

— $\phi_b(x,y) = \quad y \xrightarrow{c} x$

Cependant, remarquons que sans les arcs intérieurs étiquetés par c (en pointillés pour la figure 5.2), ce graphe appartient au niveau 2 de la hiérarchie à pile : il est substitution rationnelle inverse de la structure arborescente de la demi-droite (voir l'exemple 3.3.23 et les figures 3.15 et 3.16).

Avant d'énoncer le principal résultat de ce paragraphe (théorème 5.2.11), commençons par rappeler quelques définitions élémentaires à propos des graphes mot-automatiques.

Automates mot-automatiques. Soit Σ un alphabet and $\sharp \notin \Sigma$ un nouveau symbole. La synchronisation de deux Σ-mots, $u = a_1 \dots a_m$ et $v = b_1 \dots b_n$, est le $(\Sigma \cup \{\sharp\})^2$-mot $u \otimes v$ defini par
$u \otimes v := (a_1,b_1) \dots (a_k,b_k)(x_{k+1},y_{k+1}) \dots (x_K,y_K)$, où $k = \min(m,n)$, $K = \max(m,n)$ et pour tout $k < i \leqslant K$, $(x_i,y_i) = (\sharp,b_i)$ si $k = m$ et $(x_i,y_i) = (a_i,\sharp)$ sinon.

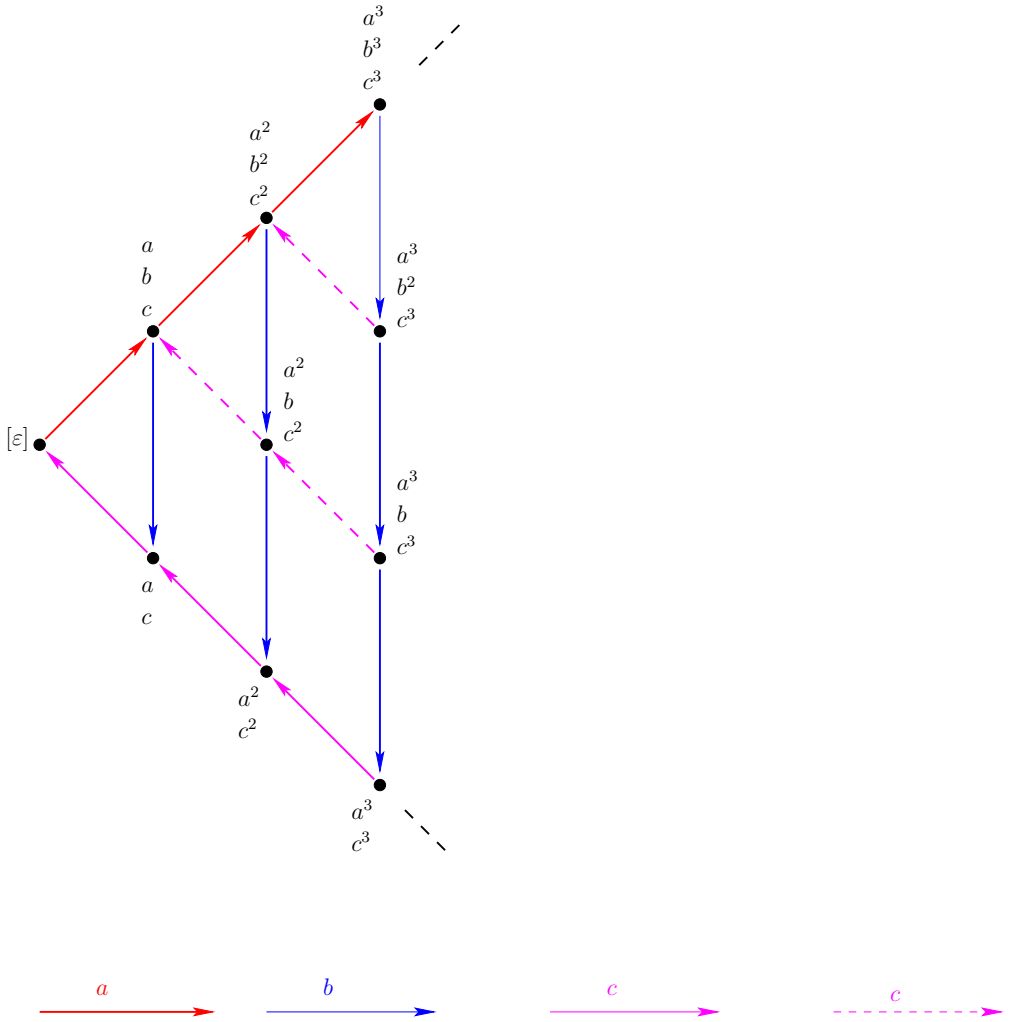


FIGURE 5.2 – Le graphe de réécriture d'un RTL (exemple 5.2.4)

Etant donné un langage L de $(\Sigma \dot\cup \{\sharp\})^2$-mots $u \otimes v$, on peut considérer la relation binaire $\{(u, v) \in \Sigma^* \times \Sigma^* \mid u \otimes v \in L\}$. Le lemme suivant montre que le domaine et l'image d'une telle relation restent des langages réguliers.

Lemme 5.2.5. *Si un langage L de $(\Sigma \dot\cup \{\sharp\})^2$-mots $u \otimes v$ est régulier, alors les langages $\{u \in \Sigma^* \mid \exists v \in \Sigma^*\ u \otimes v \in L\}$ et $\{v \in \Sigma^* \mid \exists u \in \Sigma^*\ u \otimes v \in L\}$ sont réguliers.*

La notion d'automate mot-automatique est rappelée dans la définition suivante. Intuitivement, un automate mot-automatique est un automate dont les relations binaires peuvent être codées par des langages de mots synchronisés réguliers et dont les ensembles de sommets initiaux et finaux sont réguliers également.

Définition 5.2.6. Un Λ-automate G est *mot-automatique* s'il existe un alphabet Σ, un langage régulier de Σ-mots L_{V_G} et une bijection $\nu : L_{V_G} \longrightarrow V_G$ tel que $\nu^{-1}[I_G]$

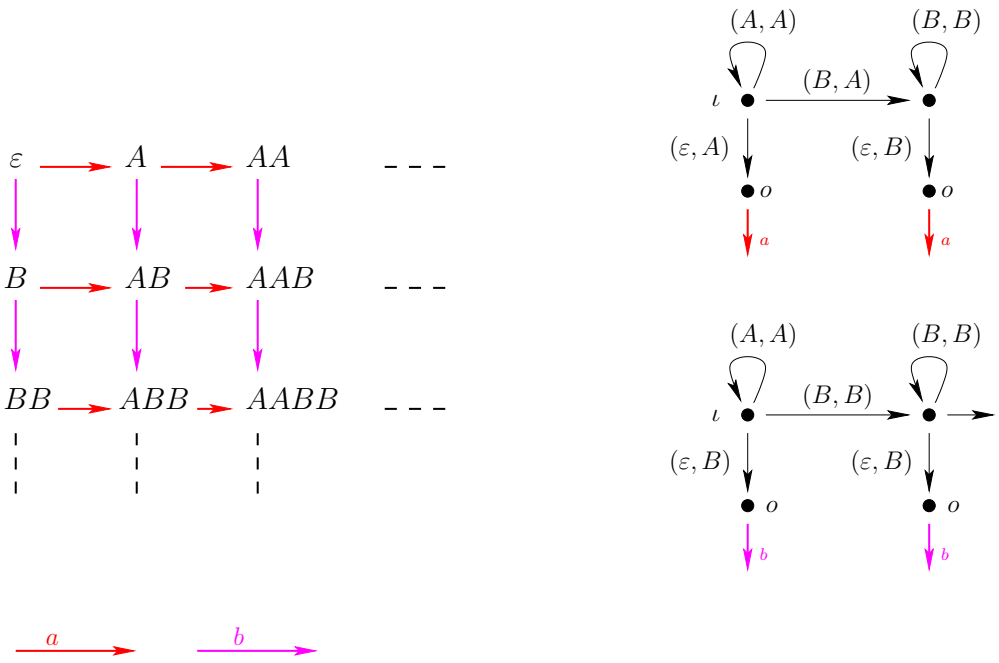


FIGURE 5.3 – Le quart de la grille infinie est mot-automatique (exemple 5.2.7). L'ensemble de ses sommets est codé par le langage A^*B^*.

et $\nu^{-1}[F_G]$ soient réguliers et pour chaque $\lambda \in \Lambda$, le langage des mots synchronisés $L_\lambda = \{\nu^{-1}(s) \otimes \nu^{-1}(t) \mid s \xrightarrow[G]{\lambda} t\}$ est régulier.

Exemple 5.2.7. Le quart de la grille infinie est mot-automatique. Voir la figure 5.3.

Le résultat suivant repose essentiellement sur les propriétés de fermeture des langages réguliers.

Théorème 5.2.8 ([21])**.** *La théorie du premier ordre d'un automate mot-automatique est décidable.*

Remarque 5.2.9. D'après le lemme 5.2.5, on peut supposer dans la définition précédente que ν est une bijection d'un langage régulier de mots sur un sur-ensemble de l'ensemble des sommets de G : un Λ-automate G est mot-automatique si et seulement s'il existe une bijection $\nu : L \longrightarrow V$, où $L \in \mathrm{Reg}(\Sigma^*)$ et $V \supseteq V_G$ telle que $\nu^{-1}[I_G]$ et $\nu^{-1}[F_G]$ soient réguliers et pour chaque $\lambda \in \Lambda$, le langage de $(\Sigma \dot\cup \{\sharp\})^2$-mots $L_\lambda = \{\nu^{-1}(s) \otimes \nu^{-1}(t) \mid s \xrightarrow[G]{\lambda} t\}$ est régulier.

Remarque 5.2.10. Le langage de $(I_D^- \dot\cup \{\sharp\})^2$-mots $\{\lceil s \rceil_{\mathbf{F}} \otimes \lceil t \rceil_{\mathbf{F}} \mid s, t \in M(\Sigma, D)\}$ est régulier. Cela résulte du lemme 4.1.4 et du fait plus général que si L_1 et L_2 désignent deux langages réguliers de mots, alors leur $\otimes$-produit $L_1 \otimes L_2 := \{u_1 \otimes u_2 \mid u_1 \in L_1, u_2 \in L_2\}$ est régulier.

Le théorème suivant résulte notamment du codage de toute trace par sa forme normale de Foata.

Théorème 5.2.11. *Un automate RTL est mot-automatique.*

Le théorème 5.2.11, que nous démontrerons ci-après, n'est plus garanti si l'on suppose que les membres gauches et droits des règles de réécriture sont réguliers par niveaux (voir la remarque 5.2.17).

Corollaire 5.2.12. *La théorie du premier ordre d'un automate RTL est décidable.*

Corollaire 5.2.13. *Le graphe de réécriture d'un système RTL est mot-automatique et sa théorie du premier ordre est décidable.*

Afin de démontrer le théorème 5.2.11, nous énonçons une propriété cruciale de compatibilité entre concaténation et forme normale de Foata.

En général, $\lceil st \rceil_{\mathbf{F}}$ et $\lceil s \rceil_{\mathbf{F}} \lceil t \rceil_{\mathbf{F}}$ peuvent être distincts. En effet, supposons que $D = \{(a,a),(b,b)\}$. Si $s = [a]$ et $t = [ab]$, alors $\lceil s \rceil_{\mathbf{F}} = \{a\}$, $\lceil t \rceil_{\mathbf{F}} = \{a,b\}$ et $\lceil st \rceil_{\mathbf{F}} = \{a,b\}\{a\}$. Cependant, le lemme suivant montre une forme de compatibilité entre concaténation et forme normale de Foata.

Lemme 5.2.14. *Soit $s,t \in M(\Sigma,D)$ tels que $\lceil s \rceil_{\mathbf{F}} = A_1 \cdots A_p$ $(p \geqslant 0)$ et $\lceil st \rceil_{\mathbf{F}} = B_1 \cdots B_m$. Alors $m \geqslant p$, $A_i \subseteq B_i$ pour chaque $1 \leqslant i \leqslant p$ et $\Pi_{I_D}((B_1 \setminus A_1) \cdots (B_p \setminus A_p) B_{p+1} \cdots B_m) = t$.*

Démonstration. Par récurrence sur la longueur de t. □

Exemple 5.2.15. Considérons la relation de dépendance

$$D = \mathrm{Id}_{\{a,b,c,d\}} \cup \{(a,b),(b,a),(a,d),(d,a)\}$$

et les traces $s = [ab]$ et $t = [cd]$. On a alors $\lceil s \rceil_{\mathbf{F}} = A_1 A_2$ avec $A_1 = \{a\}$ et $A_2 = \{b\}$. On a également $\lceil t \rceil_{\mathbf{F}} = \{c,d\}$ et $\lceil st \rceil_{\mathbf{F}} = B_1 B_2$ avec $B_1 = \{a,c\}$ et $B_2 = \{b,d\}$. Ainsi

$$\Pi_{I_D}((B_1 \setminus A_1)(B_2 \setminus A_2)) = \Pi_{I_D}(\{c\}\{d\}) = t$$

Dans la suite, pour $\lceil s \rceil_{\mathbf{F}} = A_1 \cdots A_p$ $(p \geqslant 0)$ et $t \in M(\Sigma,D)$, notons $\lceil s \rceil_{\mathbf{F}} \parallel t$ le langage de I_D-mots $B_1 \cdots B_m$ $(m \geqslant p)$ tel que $A_i \subseteq B_i$ pour chaque $1 \leqslant i \leqslant p$ et $\Pi_{I_D}((B_1 \setminus A_1) \cdots (B_p \setminus A_p) B_{p+1} \cdots B_m) = t$. Donc $\lceil st \rceil_{\mathbf{F}} \in \lceil s \rceil_{\mathbf{F}} \parallel t$, d'après le lemme ci-dessus.

Exemple 5.2.16. Supposons que $D = \{(a,a),(b,b)\}$ et considérons $s = [aba]$ et $t = [ab]$. Alors $\lceil s \rceil_{\mathbf{F}} = \{a,b\}\{a\}$ et

$$\lceil s \rceil_{\mathbf{F}} \parallel t = \{a,b\}\{a\}\varnothing^*(\{a\}\varnothing^*\{b\} + \{b\}\varnothing^*\{a\} + \{a,b\})\varnothing^* \cup \{a,b\}\{a,b\}\varnothing^*\{a\}\varnothing^*$$

Démonstration du théorème 5.2.11. Soit $\mathbf{R}$ un système reconnaissable de réécriture de traces à contextes réguliers par niveaux, I (respectivement F) un langage régulier par niveaux de traces initiales (respectivement finales). D'après la proposition 4.1.2 et le lemme 4.1.4, $\Pi_{\mathbf{F}}$ est une bijection du langage régulier $\mathbf{F}$ sur $M(\Sigma, D) \supseteq V_{\mathrm{Gr}_{\mathbf{R}}}$. Observons d'abord que comme I et F sont réguliers par niveaux, les langages de mots $\Pi_{\mathbf{F}}^{-1}[I]$ et $\Pi_{\mathbf{F}}^{-1}[F]$ sont réguliers. Il nous reste donc à montrer que pour chaque $\lambda \in \Lambda$, le langage de $(I_D \dot\cup \{\sharp\})^2$-mots

$$L_\lambda = \{\lceil [u][v] \rceil_{\mathbf{F}} \otimes \lceil [u][w] \rceil_{\mathbf{F}} \mid [u] \in \mathcal{U}, [v] \in \mathcal{V}, [w] \in \mathcal{W},\ \mathcal{U} \cdot (\mathcal{V} \xrightarrow{\lambda} \mathcal{W}) \in \mathbf{R}\}$$

est régulier.
Soit $\mathcal{U} \cdot (\mathcal{V} \xrightarrow{\lambda} \mathcal{W})$ une règle appartenant à $\mathbf{R}$. Montrons que le langage de $(I_D \dot\cup \{\sharp\})^2$-mots $\{\lceil [u][v] \rceil_{\mathbf{F}} \otimes \lceil [u][w] \rceil_{\mathbf{F}} \mid [u] \in \mathcal{U}, [v] \in \mathcal{V}, [w] \in \mathcal{W}\}$ est régulier. D'après le lemme 5.2.14 et la remarque 5.2.10 (l'intersection de deux langages régulier est un langage régulier), il suffit de montrer que le langage de $(I_D \dot\cup \{\sharp\})^2$-mots de la forme $X \otimes Y$ tel qu'il existe $[u] \in \mathcal{U}$, $[v] \in \mathcal{V}$ et $[w] \in \mathcal{W}$ tel que $X \in \lceil [u] \rceil_{\mathbf{F}} \parallel [v]$ et $Y \in \lceil [u] \rceil_{\mathbf{F}} \parallel [w]$, est régulier. Pour cela, considérons des I_D-automates $\mathcal{A}_1$, $\mathcal{A}_2$ et $\mathcal{A}_3$ (dont les uniques états initiaux sont notés respectivement $i_{\mathcal{A}_1}$, $i_{\mathcal{A}_2}$ et $i_{\mathcal{A}_3}$) qui acceptent respectivement $\{\lceil u \rceil_{\mathbf{F}} \mid [u] \in \mathcal{U}\}$, $\Pi_{I_D}^{-1}(\mathcal{V})$ et $\Pi_{I_D}^{-1}(\mathcal{W})$ et définissons le $(I_D \dot\cup \{\sharp\})^2$-automate suivant :

- l'état initial est $(i_{\mathcal{A}_1}, i_{\mathcal{A}_2}, i_{\mathcal{A}_3})$
- le $(I_D \dot\cup \{\sharp\})^2$-graphe est donné par
  $(p,q,r) \xrightarrow{A \dot\cup B / A \dot\cup C} (p',q',r') : p \xrightarrow[\mathcal{A}_1]{A} p',\ q \xrightarrow[\mathcal{A}_2]{B} q',\ r \xrightarrow[\mathcal{A}_3]{C} r'$
  $(p,q,r) \xrightarrow{B/C} (\bot,q',r') : p \in F_{\mathcal{A}_1} \cup \{\bot\},\ q \xrightarrow[\mathcal{A}_2]{B} q',\ r \xrightarrow[\mathcal{A}_3]{C} r'$
  $(p,q,r) \xrightarrow{\sharp/C} (\bot,\bot,r') : p \in F_{\mathcal{A}_1} \cup \{\bot\},\ q \in F_{\mathcal{A}_2},\ r \xrightarrow[\mathcal{A}_3]{C} r'$
  $(\bot,\bot,r) \xrightarrow{\sharp/C} (\bot,\bot,r') : r \xrightarrow[\mathcal{A}_3]{C} r'$
  $(p,q,r) \xrightarrow{B/\sharp} (\bot,q',\bot) : p \in F_{\mathcal{A}_1} \cup \{\bot\},\ r \in F_{\mathcal{A}_3},\ q \xrightarrow[\mathcal{A}_2]{B} q'$
  $(\bot,q,\bot) \xrightarrow{B/\sharp} (\bot,q'\bot) : q \xrightarrow[\mathcal{A}_2]{B} q'$
- l'ensemble des états finaux est $F = \{(p,q,r) \mid p \in F_{\mathcal{A}_1} \cup \{\bot\}, q \in F_{\mathcal{A}_2}, r \in F_{\mathcal{A}_3}\} \cup \{(\bot,\bot,r) \mid r \in F_{\mathcal{A}_3}\} \cup \{(\bot,q,\bot) \mid q \in F_{\mathcal{A}_2}\}$.

□

Remarque 5.2.17. Supposons que $D = \{(a,a),(b,b),(c,c)\}$ et considérons la règle de réécriture suivante : $[(ab)^*]([\varepsilon] \to [(bc)^*])$. Rappelons que $[(ab)^*]$ et $[(bc)^*]$ sont réguliers par niveaux mais pas reconnaissables. Le graphe de réécriture de cette règle de réécriture n'est pas mot-automatique pour le codage de l'ensemble de ses sommets par leurs formes normales de Foata. En effet, compte tenu de la proposition 5.2.5, cela impliquerait que $\Pi_{\mathbf{F}}^{-1}([(ab)^*] \cdot [(bc)^*])$ soit régulier. Ce qui n'est pas le cas (voir 4.2.3).

Nous allons montrer que l'on peut coder une machine de Minsky à deux compteurs par un automate RTL et que l'arrêt d'une telle machine est équivalent à la non-vacuité du langage accepté par celui-ci.

Proposition 5.2.18. *Le problème de la vacuité du langage accepté par un automate RTL est indécidable.*

Avant de prouver la proposition ci-dessus, faisons quelques rappels à propos des machines de Minsky à deux compteurs.

Une machine de Minsky de longueur n à deux compteurs est une suite de n instructions. La n-ième instruction est une instruction spéciale qui arrête la machine et pour chaque $k \in \{1, \ldots, n-1\}$ la k-ième instruction est de la forme

$$\left|\begin{array}{lll} k: & c := c+1; \mathrm{goto}(j) & (\mathrm{Incr}(c,j)) \\ \text{ou} & & \\ k: & \text{if } c \neq 0 \text{ then } c := c-1; \mathrm{goto}(j) \text{ else } \mathrm{goto}(l) & (\mathrm{Decr}(c,j,l)) \end{array}\right.$$

où $j, l \in \{1, \ldots, n\}$ et c est l'un des deux compteurs.

Une configuration de M est un triplet $(k, c_1, c_2) \in \{1, \ldots, n\} \times \mathbb{N} \times \mathbb{N}$, où k est le numéro de l'instruction, et c_1 et c_2 les contenus des deux compteurs. La configuration initiale est $(1, 0, 0)$. Un calcul est une suite de configurations depuis la configuration initiale et telle que deux configurations successives respectent les instructions de la machine. Le problème de l'arrêt est : étant donné une machine de Minsky à deux compteurs, un calcul fini arrête-t-il la machine ?

Théorème 5.2.19 (Minsky)**.** *Le problème de l'arrêt d'une machine de Minsky à deux compteurs est indécidable.*

Démonstration de la proposition 5.2.18. Soit M une machine de Minsky de longueur n à deux compteurs. Si c désigne l'un des deux compteurs de la machine, alors on note $\bar{c}$ l'autre compteur. Considérons alors l'automate RTL G_M défini ci-après.

- $\Sigma := \{\bot_a, \bot_b, a, b, 1, \ldots, n\}$
- La relation de dépendance D sur Σ est donnée par :

$$D = \mathrm{Id}_\Sigma \cup \{(\bot_a, a), (a, \bot_a), (\bot_b, b), (b, \bot_b)\}$$

- Pour chaque $k \in \{1, \ldots, n-1\}$ les règles de réécriture sont :
  - $[\bot_a \bot_b a^* b^*]([k] \xrightarrow{R} [cj])$ $(j \in \{1, \ldots, n\},\ c \in \{a, b\})$
    si la k-ième instruction est $\mathrm{Incr}(c, j)$
  - $[\bot_a \bot_b a^* b^*]([ck] \xrightarrow{R} [j])$ et $[\bot_{\bar{c}} \bar{c}^*]([\bot_c k] \xrightarrow{R} [\bot_c l])$
    $(j, l \in \{1, \ldots, n\},\ c \in \{a, b\})$
    si la k-ième instruction est $\mathrm{Decr}(c, j, l)$.

— L'ensemble des traces initiales est réduit au singleton $\{[\perp_a \perp_b 1]\}$.
— Les traces finales sont de la forme $[\perp_a \perp_b a^* b^* n]$.

La trace initiale code la configuration initiale de M et les traces finales codent les configurations finales. Les traces accessibles depuis la trace initiale sont de la forme $[\perp_a \perp_b a^* b^* k]$ $(k \in \{1, \ldots, n\})$. Une configuration (k, c_1, c_2) de la machine M, accessible depuis la configuration initiale, est précisément codée par la trace $[\perp_a \perp_b \overbrace{a \ldots a}^{c_1} \overbrace{b \ldots b}^{c_2} k]$. La machine M s'arrête si et seulement si

$$G_M \models \exists x \; \exists y \; (\iota(x) \; \wedge \; o(y) \; \wedge \; x \xrightarrow{*} y)$$

Autrement dit, la machine M s'arrête si et seulement si le langage accepté par G_M est non vide. □

5.3 Déplié concurrent d'un graphe concurrent

La notion d'automate fini concurrent a été introduite initialement dans [44] en tant que système de transitions asynchrone. Dans ce paragraphe, nous considérons notamment la notion de dépliage concurrent d'un graphe concurrent et nous montrons que la FO[Reach]-théorie du déplié concurrent d'un graphe fini concurrent à partir de tout sommet, est décidable. Pour ce faire, nous montrons qu'un tel déplié, enrichi de la relation d'accessibilité, est un graphe RTL. Plus généralement, nous en déduirons que la FO[Reach]-théorie de tout automate dont le graphe sous-jacent est un déplié concurrent d'un graphe fini concurrent, et dont les ensembles de sommets initiaux et finaux sont réguliers par niveaux, est décidable.

Soit (Σ, D) un alphabet de dépendance et $I = \Sigma^2 \setminus D$.

5.3.1 Graphes concurrents

Définition 5.3.1. Un Σ-graphe G est un graphe D-concurrent si
— G est un graphe déterministe
— $((a, b) \in I \text{ et } p \xrightarrow[\mathcal{A}]{ab} q) \implies p \xrightarrow[\mathcal{A}]{ba} q$.

Un Σ-automate G est D-concurrent s'il possède un unique état initial et si son graphe sous-jacent est D-concurrent.

Observation 5.3.2. Un automate concurrent est, en particulier, un automate déterministe. Réciproquement, tout automate déterministe est un automate concurrent relativement à la relation de dépendance totale sur l'alphabet d'étiquetage de ses arcs.

Observation 5.3.3. Soit $\mathcal{L} \subseteq M(\Sigma, D)$ un langage de traces. L'automate $\mathrm{Res}(\mathcal{L}, \Sigma)$ des résidus de $\mathcal{L}$ par Σ, défini par :

— le Σ-graphe $\{[u]^{-1}\mathcal{L} \xrightarrow{a} [ua]^{-1}\mathcal{L} \mid u \in \Sigma^*, a \in \Sigma\}$
— l'état initial $\mathcal{L}$
— les états finaux $[u]^{-1}\mathcal{L}$ tels que $[\varepsilon] \in [u]^{-1}\mathcal{L}$,
est un Σ-automate D-concurrent qui accepte $\bigcup \mathcal{L}$.

Exemple 5.3.4. Voir la figure 5.4 pour l'automate infini des résidus du langage non reconnaissable $[(ab)^*]$ (où a et b sont indépendants).

Exemple 5.3.5. Soit $\mathcal{L} \subseteq M(\Sigma, D)$ un langage de traces. L'automate défini par :
— le graphe de Cayley de $M(\Sigma, D)$
— l'état initial $[\varepsilon]$
— l'ensemble des états finaux $\mathcal{L}$
est un Σ-automate D-concurrent qui accepte $\bigcup \mathcal{L}$.

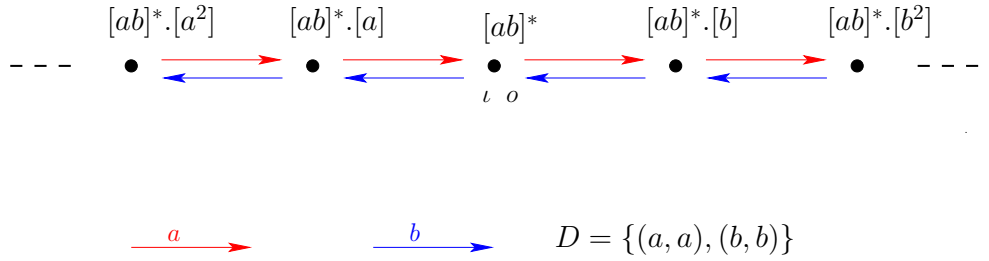


FIGURE 5.4 – $\mathrm{Res}([(ab)^*], \Sigma)$ (observation 5.3.3 et exemple 5.3.4)

En combinant la proposition 4.1.9 et l'observation 5.3.3, la reconnaissabilité d'un langage de traces correspond à celle d'un automate fini concurrent.

Proposition 5.3.6. *Un langage de traces $\mathcal{L}$ est reconnaissable si et seulement s'il existe un Σ-automate D-concurrent fini G tel que $\bigcup \mathcal{L} = L(G)$.*

5.3.2 Le déplié concurrent d'un graphe concurrent

Définition 5.3.7. Le D-déplié (concurrent) $\mathrm{Unf}_D(G, i)$ d'un Σ-graphe D-concurrent G depuis un sommet $i \in V_G$, est le Σ-graphe D-concurrent défini par :

$$\mathrm{Unf}_D(G, i) = \{[u] \xrightarrow{a} [ua] \mid u \in \Sigma^*, a \in \Sigma, i \xrightarrow[G]{ua}\}$$

Définition 5.3.8. Le D-déplié (concurrent) $\mathrm{Unf}_D(G)$ d'un Σ-automate D-concurrent G, est le Σ-automate D-concurrent défini par

$$\mathrm{Unf}_D(G) := \mathrm{Unf}_D(G, i_G) \ \cup \ \{\iota\} \times \{[\varepsilon]\} \ \cup \ \{o\} \times [L(G)]$$

L'exemple suivant permet d'observer que tout graphe de Cayley d'un monoïde de traces est déplié concurrent d'un graphe fini concurrent.

Exemple 5.3.9. Le Σ-graphe à un seul sommet

$$\{p \xrightarrow{a} p \mid a \in \Sigma\}$$

est D-concurrent. Son D-déplié concurrent depuis p est $\mathrm{Cayley}_{M(\Sigma,D)}$. Sa FO[Reach]-théorie est décidable d'après le théorème ci-dessous. En particulier, le quart de la grille infinie, $\mathrm{Cayley}_{M(\{a,b\},\mathrm{Id}_{\{a,b\}})}$, est un déplié concurrent d'un graphe fini concurrent.

Dans l'exemple suivant, on introduit l'arbre du quart de la grille infinie comme graphe de Cayley d'un certain monoïde de traces. En particulier, l'arbre du quart de la grille infinie (voir la définition 5.4.7) est déplié concurrent d'un graphe fini concurrent.

Exemple 5.3.10 (Arbre du quart de la grille infinie)**.** L'arbre du quart de la grille infinie (sur $\{a, b\}$) (voir le paragraphe 5.4) est $\mathrm{Cayley}_{M(\Sigma,D)}$ pour $\Sigma = \{a, b, c\}$ et

$$D = \mathrm{Id}_\Sigma \cup \{(a, c), (c, a), (b, c), (c, b)\}$$

Voir la figure 5.5.

Avant d'énoncer le résultat principal de ce paragraphe, rappelons que le déplié d'un graphe fini est un arbre régulier, dont la théorie du second ordre monadique est décidable (puisque le dépliage depuis un sommet MSO-définissable préserve la décidabilité de la théorie du second ordre monadique). Ici, la transformation de graphes de dépliage concurrent que nous considérons est susceptible de s'appliquer à une sur-classe stricte de celle des graphes finis déterministes.

Théorème 5.3.11. *Si G est un graphe fini D-concurrent, alors pour tout $i \in V_G$, la* FO[Reach]*-théorie de* $\mathrm{Unf}_D(G, i)$ *est décidable.*

D'une façon générale, on ne sait pas si le dépliage concurrent préserve la décidabilité de la FO[Reach]-théorie.

Démonstration du théorème 5.3.11. Considérons le $\Sigma \dot\cup \{*\}$-automate

$$\mathrm{Unf}_D(G, i)_* := \mathrm{Unf}_D(G) \cup \{[u] \xrightarrow{*} [uv] \mid u, v \in \Sigma^*, i \xrightarrow[G]{uv}\}$$

Le graphe $\mathrm{Unf}_D(G, i)_*$ est le graphe de réécriture du système reconnaissable de réécriture de traces ci-après.

$$\begin{cases} [\bigcup_{v \in V_a} L_{i,v}]([\varepsilon] \xrightarrow{a} [a]), \quad a \in \Sigma \ \text{ et } \ V_a = \{v \in V_G \mid v \xrightarrow[G]{a}\} \\ [L_{i,v}]([\varepsilon] \xrightarrow{*} [\bigcup_{w \in V_G} L_{v,w}]) \end{cases}$$

Le corollaire 5.2.12 permet de conclure. □

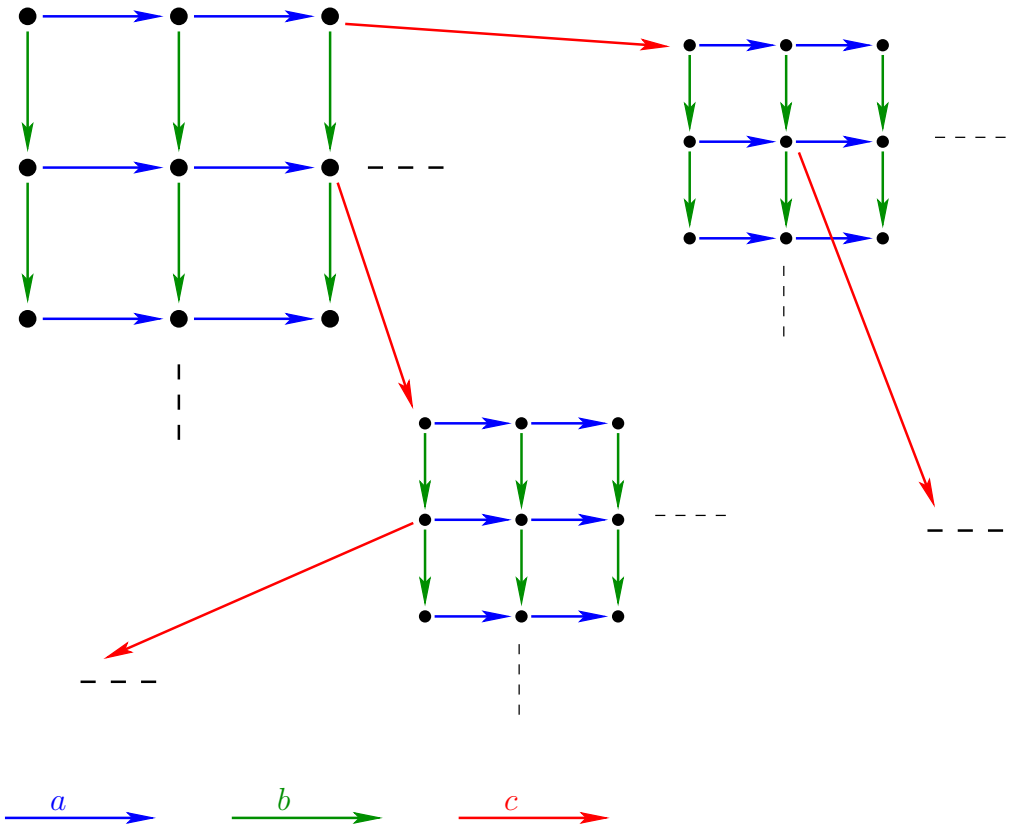


FIGURE 5.5 – L'arbre du quart de la grille infinie

Corollaire 5.3.12. *La* FO[Reach]*-théorie d'un automate dont le graphe sous-jacent est un déplié concurrent d'un graphe fini concurrent et dont les ensembles de sommets initiaux et finaux sont réguliers par niveaux, est décidable. En particulier, la* FO[Reach]*-théorie du déplié concurrent de tout automate fini concurrent est décidable.*

Démonstration. Un tel automate, muni de la relation d'accessibilité, est un automate RTL (voir la démonstration du théorème précédent). Le corollaire 5.2.12 permet de conclure. □

Remarque 5.3.13. La théorie du premier ordre avec accessibilité reconnaissable (FO[Rec]) d'un Σ-automate G est la FO-théorie de l'automate

$$G \cup \{p \xrightarrow{\mathcal{L}} q \mid p \xrightarrow{u} q, [u] \in \mathcal{L}, \mathcal{L} \in \mathrm{Rec}(M(\Sigma, D))\}$$

On peut renforcer le dernier corollaire et montrer que si G est un graphe fini D-concurrent, alors pour tout $i \in V_G$, la FO[Rec]-théorie de l'automate dont le graphe sous-jacent est $\mathrm{Unf}_D(G, i)$ et dont les ensembles de sommets initiaux et finaux sont réguliers par niveaux, est décidable. En effet, observons que chaque FO[Rec]-énoncé

ne contient qu'un nombre fini de formules atomiques $x \xrightarrow{\mathcal{L}_1} y$,..., $x \xrightarrow{\mathcal{L}_n} y$ $(n \geqslant 1)$. Et

$$\mathrm{Unf}_D(G,i) \cup \{p \xrightarrow{\mathcal{L}_j} q \mid p \xrightarrow{u} q, [u] \in \mathcal{L}_j, j \in \{1,\ldots,n\}\}$$

est le graphe de réécriture du RTL donné par l'ensemble des règles de réécriture ci-après.

$$\begin{cases} [\bigcup_{v \in V_a} L_{i,v}] \left([\varepsilon] \xrightarrow{a} [a]\right), \quad a \in \Sigma \ \text{ et } \ V_a = \{v \in V_G \mid v \xrightarrow[G]{a}\} \\ [L_{i,v}] \left([\varepsilon] \xrightarrow{\mathcal{L}_j} [\bigcup_{w \in V_G} L_{v,w}] \cap \mathcal{L}_j\right), \quad j \in \{1,\ldots,n\}. \end{cases}$$

Compte tenu de l'exemple 5.3.9, observons que la décidabilité de la FO[Rec]-théorie du graphe de Cayley d'un monoïde de traces se déduit de la décidabilité de la FO-théorie d'un graphe RTL. La remarque suivante montre la réduction inverse.

Remarque 5.3.14. Tout graphe de réécriture d'un système RTL (à contextes reconnaissables) sur un monoïde de traces $M(\Sigma, D)$ est FO[Rec]-interprétation du graphe de Cayley de ce monoïde de traces. En effet, observons que l'élément neutre est FO-définissable :

$$\mathrm{neutral}(x) = \bigwedge_{a \in \Sigma} \neg\ \exists t\ t \xrightarrow{a} x$$

Ensuite, pour chaque règle de la forme $\mathcal{U} \cdot (\mathcal{V} \to \mathcal{W})$ (où $\mathcal{U}, \mathcal{V}, \mathcal{W} \in \mathrm{Rec}(M(\Sigma, D))$), il suffit de considérer la formule

$$\phi(x,y) = \exists i\ \exists z\ (\mathrm{neutral}(i)\ \wedge\ i \xrightarrow{\mathcal{U}} z \wedge z \xrightarrow{\mathcal{V}} x \wedge z \xrightarrow{\mathcal{W}} y)$$

5.3.3 Structure d'événements trace-régulière

Dans [25], Madhusudan montre que la FO-théorie d'une structure d'événements trace-régulière est décidable. Pour cela, il montre que l'ensemble des sommets et les relations d'un tel graphe peuvent être codés par un langage de traces sur un alphabet de dépendance judicieux. Notons que compte-tenu de la niveau-régularité des contextes, cette technique ne permet pas de montrer que la FO-théorie d'un graphe RTL est décidable.

Une trace $t = [a_1 \cdots a_n] \in M(\Sigma, D)$ est *première* si l'ensemble $\{1,\ldots,n\}$, partiellement ordonné par la relation E définie par iEj si et seulement si $i < j$ et $a_i D a_j$, possède exactement un élément maximal.

Soit $\mathcal{L} \subseteq M(\Sigma, D)$ un langage de traces. Notons $\mathrm{prime}(\mathcal{L})$ l'ensemble des traces premières appartenant à $\mathcal{L}$.

Définition 5.3.15. La structure d'évènements $\mathcal{ES}_{\mathcal{L}}$ définie par $\mathcal{L}$ est le $\{\leqslant, \sharp, (\lambda_a)_{a \in \Sigma}\}$-graphe sur $\mathrm{prime}(\mathcal{L})$ défini par :

— $t \xrightarrow{\leqslant} t' : t \sqsubseteq t'$

— $t \xrightarrow{\sharp} t' : \forall t'' \in \text{prime}(\mathcal{L})(t \not\sqsubseteq t'' \vee t' \not\sqsubseteq t''))\}$
— $t \xrightarrow{\lambda_a} t$: l'élément maximal de t est a.

Théorème 5.3.16 ([25])**.** *Si $\mathcal{L} \in \text{Rec}(M(\Sigma, D))$, alors la* FO*-théorie de $\mathcal{ES}_{\mathcal{L}}$ est décidable.*

Démonstration. Une trace $t \in V_{\text{Unf}_D(\text{Res}(\mathcal{L},\Sigma))_*}$ est première si et seulement si t n'est pas le successeur de deux sommet distincts de $\text{Unf}_D(\text{Res}(\mathcal{L}, \Sigma))_*$. Puisque cette dernière propriété est FO-définissable, la structure d'évènements $\mathcal{ES}_{\mathcal{L}}$ peut être obtenue par FO interpretation de $\text{Unf}_D(\text{Res}(\mathcal{L}, \Sigma))_*$, dont la FO-théorie est décidable. □

5.4 Arbre de graphe

Dans ce paragraphe, nous considérons les graphes de réécriture suffixe de systèmes de réécriture de termes clos (graphe GTR). La FO[Reach]-théorie d'un tel graphe est décidable [17]. Nous définissons une notion d'arbre de graphe et nous prouvons que si l'arbre d'un graphe est un graphe GTR, alors il est finiment décomposable par taille. Par conséquent, l'arbre du quart de la grille infinie, défini comme le déplié concurrent d'un graphe fini concurrent (exemple 5.5), n'est pas un graphe GTR, bien que sa FO[Reach]-théorie soit décidable.

5.4.1 Graphes de réécriture suffixe d'un système de réécriture de termes clos (graphes GTR)

Une *position* est un élément de l'ensemble $\mathbb{N}^*$ des mots finis sur $\mathbb{N}$. Notons $\sqsubseteq$ l'ordre préfixe sur $\mathbb{N}^*$. Soit F un alphabet gradué (chaque symbole dans F possède une arité appartenant à $\mathbb{N}$). Un *terme* t sur F est une fonction partielle $t : \mathbb{N}^* \longrightarrow F$ dont le domaine, $\text{Dom}(t)$, a les propriétés suivantes :

— $\text{Dom}(t) \neq \varnothing$
— $\text{Dom}(t)$ est fermé par préfixe
— $\forall u \in \text{Dom}(t)$, si l'arité de $t(u)$ est n $(n \geqslant 0)$, alors $\{j \mid uj \in \text{Dom}(t)\} = \{1, \ldots, n\}$.

La *taille* d'un terme t est le nombre de ses noeuds, c'est-à-dire le cardinal de $\text{Dom}(t)$. Le *sous-terme* de t à la position u, notée $t{\downarrow}\, u$, est le terme sur F défini par :

— $\text{Dom}(t{\downarrow}\, u) = \{v \in \mathbb{N}^* \mid uv \in \text{Dom}(t)\}$
— $\forall v \in \text{Dom}(t{\downarrow}\, u)$, $(t{\downarrow}\, u)(v) = t(uv)$.

Si $u \in \text{Dom}(t)$ et s est un terme, alors $t[u \leftarrow s]$ est le terme obtenu à partir de t en remplaçant le sous-terme $t{\downarrow}\, u$ par s :

$$t[u \leftarrow s](v) = \begin{cases} s(w) \text{ si } v = uw \text{ et } w \in \text{Dom}(s) \\ t(v) \text{ si } v \in \text{Dom}(t) \text{ et } u \not\sqsubseteq v \end{cases}$$

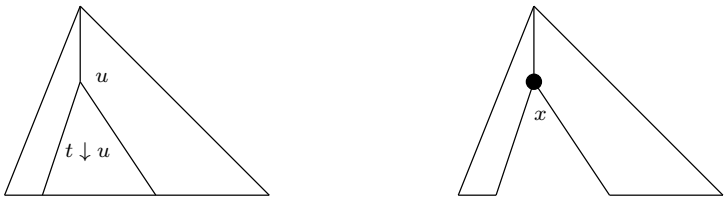


FIGURE 5.6 – Le contexte d'un terme t à la position u

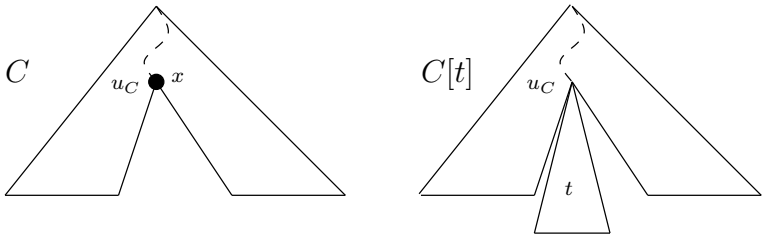


FIGURE 5.7 – Un contexte

Si t est un terme sur F et $u \in \mathrm{Dom}(t)$, alors le *contexte de t à la position u* est le terme $t[u \leftarrow x]$ sur $F \dot\cup \{x\}$, où x est une constante (*i.e.* l'arité de x est 0).

Un *contexte* C sur F est un terme sur $F \dot\cup \{x\}$, x constante, tel qu'il existe une unique position $u_C \in \mathrm{Dom}(C)$ pour laquelle $C(u_C) = x$. Si t est un terme sur F, alors le terme $C[t]$ sur F est défini par $C[t] := C[u_C \leftarrow t]$. La taille $|C|$ d'un contexte C sur F est le nombre de ses noeuds moins 1 (voir les figures 5.6 et 5.7).

Un *système de réécriture de termes clos* $\mathbf{R}$ est un quadruplet $\mathbf{R} = (F, \Sigma, R, i)$, où :

— F est un alphabet gradué
— Σ est un alphabet (d'étiquetage)
— $R := \bigcup_{a \in \Sigma} R_a$, où pour chaque $a \in \Sigma$, R_a est un ensemble fini de règles (de réécriture) de la forme $s \xrightarrow{a} s'$ avec s et s' des termes distincts sur F
— i est un terme initial sur F.

On écrit :

— $t \xrightarrow[\mathbf{R}]{a} t'$ s'il existe une position $p \in \mathrm{Dom}(t)$ et une règle $s \xrightarrow{a} s' \in R_a$ tel que $t{\downarrow} p = s$ et $t' = t[p \leftarrow s']$
— $t \xrightarrow[\mathbf{R}]{} t'$ lorsqu'il existe $a \in \Sigma$ tel que $t \xrightarrow[\mathbf{R}]{a} t'$
— $\xrightarrow[\mathbf{R}]{*}$ pour la clôture réflexive et transitive de $\xrightarrow[\mathbf{R}]{}$.

Le *graphe suffixe* $\mathrm{Gr}_{\mathbf{R}}$ de $\mathbf{R}$ est le Σ-graphe défini par

$$\mathrm{Gr}_{\mathbf{R}} := \{t \xrightarrow[\mathbf{R}]{a} t' \mid i \xrightarrow[\mathbf{R}]{*} t, a \in \Sigma\}$$

Un graphe est dit graphe suffixe d'un système de réécriture de termes clos (graphe GTR) s'il est isomorphe au graphe suffixe d'un système de réécriture de termes clos.

Remarque 5.4.1. Les graphes GTR n'ont pas de boucle puisque les membres gauches et droits de chaque règle appartenant au système de réécriture sont distincts.

Exemple 5.4.2. Soit l'alphabet gradué $F := \{a, b\}$, où le symbole a (respectivement le symbole b) est d'arité 0 (respectivement d'arité 2). On considère le système de réécriture de termes clos $(F, \{r\}, R, a)$, où R n'est constitué que de la seule règle $a \xrightarrow{r} b(a, a)$. Voir la figure 5.8 pour son graphe suffixe. Observons qu'il n'est pas de degré borné.

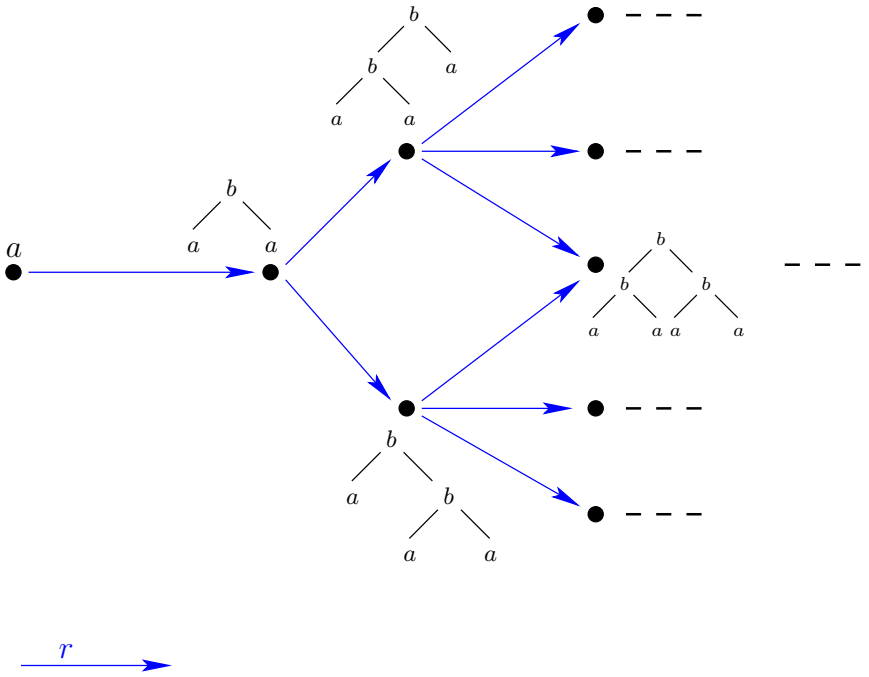


FIGURE 5.8 – Un graphe GTR dont le degré n'est pas borné.

Exemple 5.4.3. Le quart de la grille infinie est un graphe GTR (voir la figure 5.9).

Dans [17], Dauchet et Tison montrent qu'un graphe GTR muni de la relation d'accessibilité est arbre automatique. La FO[Reach]-théorie d'un tel graphe est donc décidable.

Théorème 5.4.4 ([17])**.** *La* FO[Reach]*-théorie d'un graphe GTR est décidable.*

5.4.2 Décomposition finie d'un graphe

On commence par rappeler la définition de la frontière d'un sous-graphe (relativement au graphe dont il est une partie).

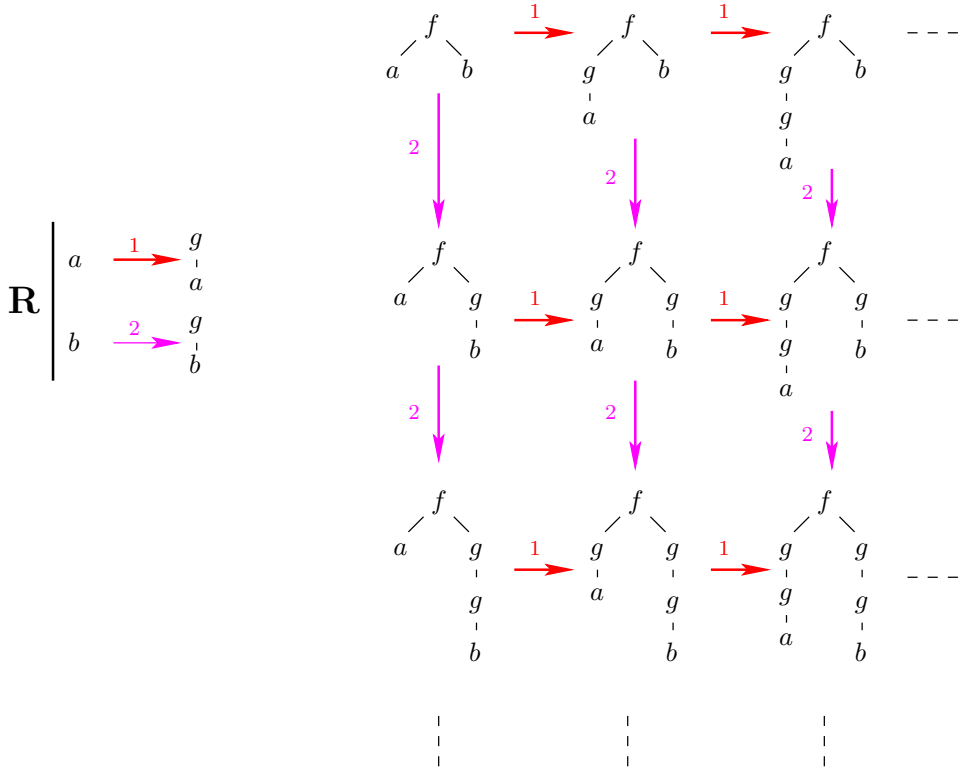


FIGURE 5.9 – Le quart de grille infinie est un graphe GTR.

Définition 5.4.5. Soit G un graphe et $H \subseteq G$ un sous-graphe de G. La frontière de H (dans G) est $\mathrm{Fr}(H) = V_H \cap V_{G-H}$.

La frontière de H est l'ensemble des sommets de H qui sont incidents à un arc dans $G - H$ (voir la figure 5.10). En particulier, $\mathrm{Fr}(H) = \mathrm{Fr}(G - H)$.

Soit $\mathrm{Gr}_{\mathbf{R}}$ un graphe GTR. Pour chaque $n \geqslant 0$, on note

$$G_n := \{s \xrightarrow{e} t \in \mathrm{Gr}_{\mathbf{R}} \mid |s| < n \text{ ou } |t| < n\}$$

Selon la définition 5.4.5, la frontière de $\mathrm{Gr}_{\mathbf{R}} - G_n$ est

$$\mathrm{Fr}(G_n) = \{s \in V_{\mathrm{Gr}_{\mathbf{R}}} \mid |s| \geqslant n \ \wedge \ \exists\, r, t\ (|r| < n \leqslant |t| \ \wedge \ r \underset{G_n}{\text{---}} s \underset{\mathrm{Gr}_{\mathbf{R}}}{\text{---}} t)\}$$

Voir la figure 5.11.

Et la frontière d'une composante connexe K de $\mathrm{Gr}_{\mathbf{R}} - G_n$ est

$$\mathrm{Fr}(K) = \mathrm{Fr}(\mathrm{Gr}_{\mathbf{R}} - G_n) \cap V_K$$

La frontière de K est constituée des sommets de K incidents à un arc de G_n.

Le graphe $\mathrm{Gr}_{\mathbf{R}}$ est *finiment décomposable par taille* si

$$\mathrm{dec} := \{(K, \mathrm{Fr}(K)) \mid K \text{ composante connexe de } \mathrm{Gr}_{\mathbf{R}} - G_n, n \geqslant 0\}$$

est d'indice fini pour la relation d'isomorphisme.

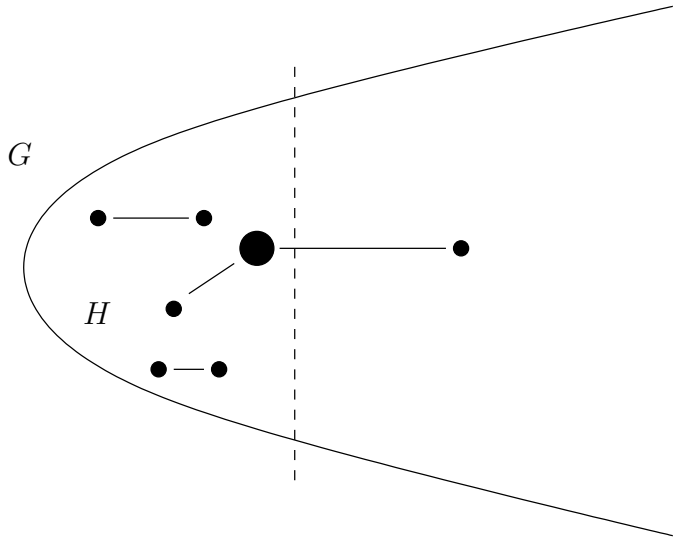


FIGURE 5.10 – La frontière d'un sous-graphe H de G.

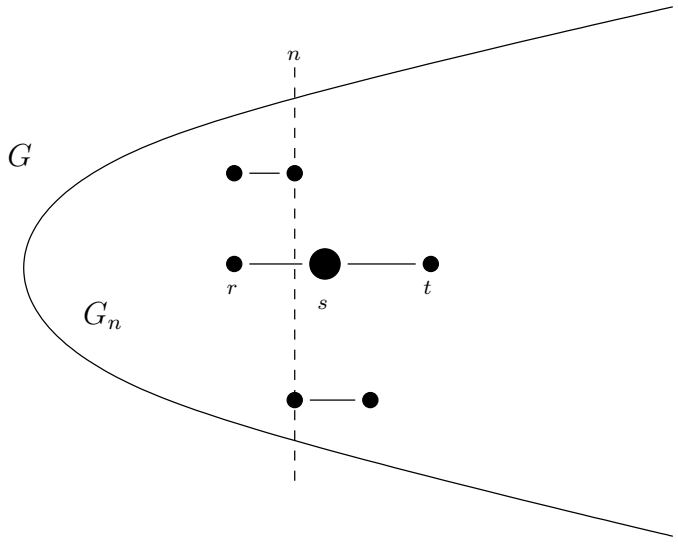


FIGURE 5.11 – La frontière de G_n dans $\mathrm{Gr}_{\mathbf{R}}$.

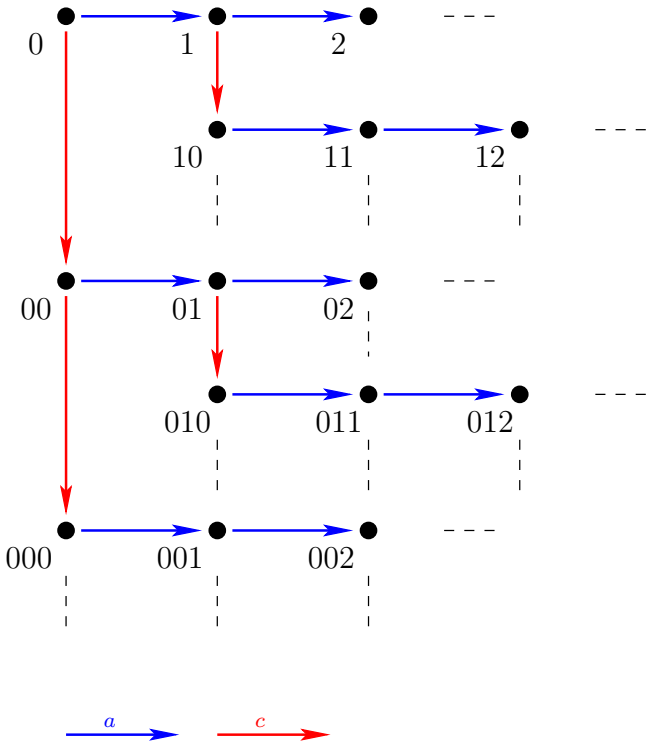


FIGURE 5.12 – L'arbre de la demi-droite.

Théorème 5.4.6 ([7]). *Si un graphe est finiment décomposable par taille, alors il appartient au premier niveau de la hiérarchie à pile. En particulier, sa MSO-théorie est décidable.*

5.4.3 Arbre de graphe et décomposition finie

Définition 5.4.7. Soit G un Σ-graphe et $p_0 \in V_G$. Etant donné un nouveau symbole $c \notin \Sigma$, l'arbre de G à partir de p_0 est le $\Sigma \dot\cup \{c\}$-graphe $\mathrm{Tree}(G, p_0)$ défini par

$$\mathrm{Tree}(G, p_0) := \{up \xrightarrow{a} uq \mid u \in V_G^*, p \xrightarrow[G]{a} q\} \cup \{u \xrightarrow{c} up_0 \mid u \in V_G^*\}$$

Exemple 5.4.8. Voir la figure 5.12 pour l'arbre de la demi-droite.

Remarque 5.4.9. Le graphe $\mathrm{Tree}(G, p_0)$ est c-déterministe :

$$(v \xrightarrow[\mathrm{Tree}(G,p_0)]{c} v_1 \text{ et } v \xrightarrow[\mathrm{Tree}(G,p_0)]{c} v_2) \Longrightarrow v_1 = v_2$$

Remarque 5.4.10. Le graphe $\mathrm{Tree}(G, p_0)$ est un arbre si et seulement si G est un arbre.

Remarque 5.4.11. Soit (Σ, D) un alphabet de dépendance, G un Σ-graphe fini D-concurrent et $p_0 \in V_G$. Le $\Sigma \dot\cup \{c\}$-graphe défini par $G \cup \{p \xrightarrow{c} p_0 \mid p \in V_G\}$ est D_c-concurrent, avec $D_c = D \cup \{(a,b) \mid a,b \in \Sigma \cup \{c\} \wedge (a = c \vee b = c)\}$. Son D_c-déplié à partir de p_0 est $\mathrm{Tree}(\mathrm{Unf}_D(G,p_0),p_0)$.

Théorème 5.4.12. *Si* $\mathrm{Tree}(G,p_0)$ *est un graphe GTR, alors* $\mathrm{Tree}(G,p_0)$ *est finiment décomposable par taille.*

La MSO-théorie du quart de la grille infinie est indécidable. C'est aussi le cas de la MSO-théorie de l'arbre de ce dernier. En combinant le théorème 5.4.12 et le théorème 5.4.6, on en déduit le corollaire ci-dessous.

Corollaire 5.4.13. *L'arbre du quart de la grille infinie n'est pas un graphe GTR.*

5.4.4 Démonstration du théorème 5.4.12

Lemme 5.4.14. *Soit G un Σ-graphe et $p_0 \in V_G$. S'il existe un système de réécriture de termes clos $\mathbf{R} = (F, \Sigma, R, i)$ tel que* $\mathrm{Tree}(G,p_0)$ *soit isomorphe à* $\mathrm{Gr}_{\mathbf{R}}$*, alors pour tout terme $t \in V_{\mathrm{Gr}_{\mathbf{R}}}$, il existe une plus petite position u_t (pour l'ordre préfixe $\sqsubseteq$) à partir de laquelle t est incident à une réécriture dans* $\mathrm{Gr}_{\mathbf{R}}$.

Démonstration du lemme 5.4.14. Il suffit de montrer que s'il existe des positions incomparables u' et u'' à partir desquelles t est incident à des réécritures, alors il existe une position v, où $v \sqsubseteq u'$ et $v \sqsubseteq u''$, à partir de laquelle t est incident à une réécriture.

Notons e' (respectivement e'') l'étiquette de la réécriture dont t est incident à partir de la position u' (respectivement u''). Rappelons que

$$\mathrm{Tree}(G,p_0) := \{up \xrightarrow{a} uq \mid u \in V_G^*, p \xrightarrow[G]{a} q\} \cup \{u \xrightarrow{c} up_0 \mid u \in V_G^*\}$$

où $c \notin \Sigma$. Nous allons montrer que $c \notin \{e', e''\}$. Puisque u' et u'' sont incomparables et compte tenu de la remarque 5.4.1, il existe deux chemins entre deux sommets distincts de $\mathrm{Gr}_{\mathbf{R}}$, étiquetés par $e'e''$ et $e''e'$, chacun d'eux sans boucle (voir la figure 5.13). Cela n'est possible dans $\mathrm{Tree}(G,p_0)$ que si $c \notin \{e', e''\}$. En effet,

- $\{e', e''\} \subseteq \{c\}$ est impossible à cause de la remarque 5.4.9
- $e' = c$ et $e'' \in \Sigma$ (ou l'inverse $e'' = c$ et $e' \in \Sigma$) est impossible car c et e'' ne commutent pas dans $\mathrm{Tree}(G,p_0)$.

Mais il existe une position v à partir de laquelle le terme t est incident à une réécriture etiquetée par c. Compte tenu du point précédent, la position v doit être comparable aux positions u' et u''. Puisque u' et u'' ne sont pas comparables, on en déduit que $v \sqsubseteq u'$ et $v \sqsubseteq u''$. □

Démonstration du théorème 5.4.12. Soit $\mathbf{R} = (F, \Sigma, R, i)$ un système de réécriture de termes clos tel que $\mathrm{Tree}(G,p_0)$ soit isomorphe à $\mathrm{Gr}_{\mathbf{R}}$. Nous devons montrer que

$$\mathrm{dec} := \{(K, V_K \cap V_{G_n}) \mid K \text{ composante connexe de } \mathrm{Gr}_{\mathbf{R}} - G_n, n \geqslant 0\}$$

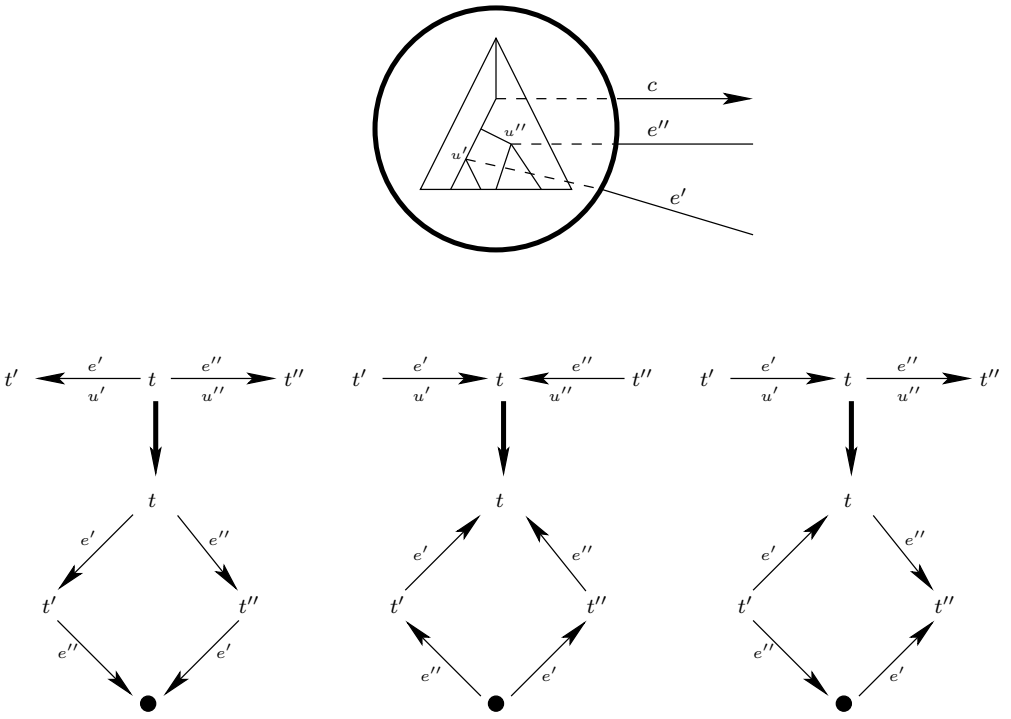


FIGURE 5.13 – Chemins dans $\mathrm{Gr}_{\mathbf{R}}$ entre deux sommets distincts, étiquetés par $e'e''$ et $e''e'$

est d'indice fini. Soit $\delta := \max\{||d| - |g|| \mid g \xrightarrow{e} d \in R\}$ et $M := \max\{|g|, |d| \mid g \xrightarrow{e} d \in R\}$. Nous allons montrer que pour chaque composante connexe K dans dec, il existe une position u_K et un contexte C_K tels que

— pour tout terme $t \in V_K$, $u_K \in \mathrm{Dom}(t)$ et C_K est le contexte de t à la position u_K ($t = C_K[t{\downarrow}\, u_K]$)

— pour tout terme $t \in \mathrm{Fr}_{\mathrm{Gr}_{\mathbf{R}}}(K)$, $|t{\downarrow}\, u_K| < M + \delta$.

Ensuite la partie finie de l'ensemble fini des termes de taille au plus $M + \delta$, obtenue à partir de $\mathrm{Fr}_{\mathrm{Gr}_{\mathbf{R}}}(K)$ en supprimant le contexte C_K, est caractéristique du type d'isomorphie de $(K, \mathrm{Fr}_{\mathrm{Gr}_{\mathbf{R}}}(K))$. En effet, pour $K \in$ dec, soit $\widetilde{K} := \{s \mid C_K[s] \in \mathrm{Fr}_{\mathrm{Gr}_{\mathbf{R}}}(K)\}$. Si $\widetilde{K} = \widetilde{K'}$, alors $(K, \mathrm{Fr}_{\mathrm{Gr}_{\mathbf{R}}}(K))$ et $(K', \mathrm{Fr}_{\mathrm{Gr}_{\mathbf{R}}}(K'))$ sont isomorphes *via* $C_K[s] \mapsto C_{K'}[s]$.

Soit $K \in$ dec et $n \geqslant 0$ tels que K soit une composante connexe de $\mathrm{Gr}_{\mathbf{R}} - G_n$. Remarquons que pour tout $t \in \mathrm{Fr}_{\mathrm{Gr}_{\mathbf{R}}}(K)$, $n \leqslant |t| < n + \delta$. En particulier, $\mathrm{Fr}_{\mathrm{Gr}_{\mathbf{R}}}(K)$ est fini.

Soit $m_K := \min\{|t| \mid t \in V_K\}$. Donc $n \leqslant m_K$. Considérons $t_K \in V_K$ tel que $|t_K| = m_K$. Puisque t_K n'est pas un sommet isolé dans K, il existe une position à partir de laquelle t_K est incident à une réécriture dans K. Soit u_K le plus petit préfixe de cette position tel que $|t_K{\downarrow}\, u_K| \leqslant M$. Le terme t_K peut être écrit $t_K = C_K[t_K{\downarrow}\, u_K]$, avec C_K un contexte.

Nous allons montrer que tout terme $t \in V_K$ est défini à la position u_K et que le contexte de t à la position u_K est C_K. Il suffit de montrer l'affirmation suivante.

Affirmation 1. *Soit $t \in V_K$. La position u_K est un préfixe de toute position à partir de laquelle le terme t est incident à une réécriture dans K.*

Soit $t \in \mathrm{Fr}_{\mathrm{Gr}_{\mathbf{R}}}(K)$. Rappelons que $n \leqslant |t| < n + \delta$. Puisque $|t{\downarrow}\, u_K| = |t| - |C_K|$, on en déduit que $|t{\downarrow}\, u_K| < n + \delta - |C_K| \leqslant m_K + \delta - |C_K|$. Mais $m_K - |C_K| = |t_K{\downarrow}\, u_K| \leqslant M$. Par suite $|t{\downarrow}\, u_K| < M + \delta$.

□

Démonstration de l'affirmation 1. Supposons (comme c'est le cas pour le terme t_K) qu'il existe une position u à partir de laquelle un terme t est incident à une réécriture dans K telle que $u_K \sqsubseteq u$ et que le contexte de t à la position u_K soit C_K. Nous allons montrer que si v est une position à partir de laquelle t est incident à une réécriture dans K, alors $u_K \sqsubseteq v$. Puisque K est connexe, l'affirmation sera prouvée.

D'abord, remarquons qu'il n'existe pas de position p plus petite que u_K à partir de laquelle t est incident à une réécriture (dans $\mathrm{Gr}_{\mathbf{R}}$) : puisque $t \in V_K$ et t_K (qui est de taille minimale dans V_K) ont le même contexte C_K, on a $|t{\downarrow}\, u_K| \geqslant |t_K{\downarrow}\, u_K|$ et donc $|t{\downarrow}\, p| \geqslant |t_K{\downarrow}\, p| > M$. On en déduit que s'il existe une position p à partir de laquelle t est incident à une réécriture et telle que p et u sont comparables, alors $u_K \sqsubseteq p$.

Ensuite, considérons la plus petite position u_t à partir de laquelle le terme t est incident à une réécriture (lemme 5.4.14). D'après le point précédent, on a $u_K \sqsubseteq u_t \sqsubseteq u$. Donc $u_K \sqsubseteq v$. □

5.5 Conclusion

Nous avons montré qu'un automate RTL est mot-automatique et en avons déduit que sa théorie du premier ordre est décidable. Nous avons aussi montré que le problème de la vacuité du langage accepté par un tel automate n'est pas décidable. De plus, nous avons montré que la FO[Reach]-théorie du déplié concurrent d'un automate fini concurrent est décidable. Enfin, nous avons montré que la classe des dépliés concurrents des graphes finis concurrents n'est pas incluse dans celle des graphes GTR. En particulier, la classe des dépliés concurrents des graphes finis concurrents constitue une classe de DAG mot-automatiques non incluse dans celle des graphes GTR et dont la FO[Reach]-théorie est décidable.

En résumé, nous avons étendu le premier niveau de la hiérarchie à pile qui est constitué des graphes de réécriture des systèmes reconnaissables de réécriture de mots aux graphes RTL. Rappelons que les graphes du premier niveau de la hiérarchie à pile sont les interprétations monadiques des arbres réguliers (*i.e.*, les dépliés des graphes finis) et qu'un arbre régulier déterministe n'est que le déplié concurrent d'un graphe fini déterministe pour la relation de dépendance totale. Nous avons observé

qu'un graphe RTL est FO[Rec]-interprétation du graphe de Cayley du monoïde de traces sous-jacent (voir la remarque 5.3.14) qui est lui même le déplié concurrent d'un graphe fini concurrent. Deux questions se posent. D'abord, on peut se demander si réciproquement, une FO[Rec]-interprétation d'un déplié concurrent d'un graphe fini concurrent est un graphe RTL, alors que l'analogue de cette réciproque est vraie dans le cas des mots. Ensuite, on peut se demander si la transformation de dépliage concurrent préserve la décidabilité de la FO[Reach]-logique.

Chapitre 6

Algèbres de Boole de langages de mots

6.1 Introduction

Une algèbre de Boole de langages de mots sur un alphabet donné est une famille de langages fermée par intersection et par complémentation. La mise en évidence d'algèbres de Boole de langages est notamment utile en vérification automatique. Par exemple, le fait que la théorie du premier ordre de tout graphe mot-automatique est décidable repose sur les propriétés de fermeture des langages réguliers par projection et par les opérations booléennes : à toute relation définie par une formule du premier ordre correspond de manière effective un langage régulier et le problème de la satisfaction par le graphe d'un énoncé donné se réduit au problème de la vacuité du langage régulier correspondant. S'il est bien connu que la famille des langages réguliers ou celle des langages contextuels forme une algèbre de Boole, il est aussi bien connu que ce n'est pas le cas de la famille des langages algébriques. Toutefois, diverses sous-familles de langages algébriques formant une algèbre de Boole ont été mises en évidence [30, 35, 6], comme par exemple l'algèbre de Boole des langages visibles relativement à un alphabet visible [1].

Dans ce chapitre, nous considérons des automates de traces. Leurs sommets sont des traces de Mazurkiewicz [18] mais les langages acceptés restent bien des langages de mots. Définissant la longueur d'une trace comme étant la longueur de sa forme normale de Foata, nous montrons comment obtenir diverses algèbres de Boole à partir de toute famille $\mathcal{F}$ d'automates de traces fermée par deux opérations, que sont la synchronisation par niveaux et la superposition par niveaux (théorème 6.3.17). Ces opérations garantissent la fermeture respectivement par intersection et complémentation des langages acceptés. Plus précisément, nous montrons que pour tout automate déterministe $H \in \mathcal{F}$, la classe des langages acceptés par les automates déterministes appartenant à $\mathcal{F}$ et qui sont longueur-réductibles en H forment une algèbre de Boole de langages (de mots).

Ensuite nous appliquons le théorème précédent à la famille TrSuffix des automates suffixes de traces à contextes réguliers par niveaux. Nous montrons que cette famille satisfait les conditions de fermeture énoncées ci-dessus.

Enfin, nous montrons que la sous-famille $\mathrm{TrSuffix}^{\mathrm{Petri}}$ des automates suffixes de traces sur des monoïdes de traces dont la relation de dépendance est l'égalité satisfont aussi les conditions de fermeture énoncées ci-dessus. Nous appelons réseau de Petri généralisé un tel automate. Il s'agit d'une notion de réseau de Petri plus générale que celle présentée dans le deuxième chapitre. En effet, si dans un réseau de Petri usuel, une transition (v_1, v_2) (v_1 et v_2 vecteurs d'entiers naturels de même dimension p) peut s'appliquer à tout vecteur d'entiers naturels v (de dimension p), pourvu que $v - v_1$ reste un vecteur d'entiers naturels, pour la notion de réseau de Petri généralisé introduite, ce ne sera pas le cas. En effet, ici, un vecteur d'entiers naturels est codé par une trace (sur un alphabet de dépendance dont la relation de dépendance est l'égalité) et celle-ci ne sera source d'une transition que si l'un de ses préfixes appartient à un certain contexte (un langage de traces) qui sera supposé régulier par niveaux. Un réseau de Petri généralisé sera également muni d'un ensemble régulier par niveaux de sommets finaux. Nous obtenons donc diverses algèbres de Boole de langages de mots acceptés par de tels réseaux de Petri généralisés déterministes.

Les résultats de ce chapitre ont été publiés dans [26].

6.2 Préliminaires

6.2.1 Morphisme d'automates

Soit T un alphabet.

Un *morphisme* f d'un T-automate G dans un T-automate H est une application $f : V_G \to V_H$ telle que

$$u \xrightarrow[G]{a} v \Longrightarrow f(u) \xrightarrow[H]{a} f(v) \quad \text{et} \quad (c, u) \in G \Longrightarrow (c, f(u)) \in H \quad (c \in \{\iota, o\})$$

Si un tel morphisme existe, nous écrivons $G \xrightarrow{f} H$ (ou plus simplement $G \to H$) et nous disons que G est (f-)*reductible* dans H. Les automates G et H sont f-*isomorphes* s'il existe une bijection $f : V_G \to V_H$ telle que $G \xrightarrow{f} H$ et $H \xrightarrow{f^{-1}} G$.

Lemme 6.2.1. *Soient G et H des automates. Si $G \to H$, alors $L(G) \subseteq L(H)$.*

On fixe une application $|\ | : V \longrightarrow \mathbb{N}$, qu'on appelle longueur. Un morphisme f *préserve la longueur* si $|f(v)| = |v|$ pour n'importe quel $v \in V_G$. Si un tel morphisme existe, nous disons que G est *longueur-réductible* dans H et nous écrivons $G \xrightarrow{f}_{\ell} H$ (ou plus simplement $G \to_{\ell} H$). Les automates G et H sont *longueur-isomorphes* s'il existe un morphisme f préservant la longueur et tel que G et H soient f-isomorphes.

Exemple 6.2.2. Considérons l'arbre binaire infini

$$T_2 := \{Ku \xrightarrow{a} Kua \mid u \in \{a,b\}^*\} \ \cup \ \{Ku \xrightarrow{b} Kub \mid u \in \{a,b\}^*\} \\ \cup \ \{(\iota, K)\} \ \cup \ \{o\} \times K\{a,b\}^*$$

et l'automate (visible)

$$\mathrm{Vis}(\{c\}, \varnothing, \{a,b\}) := \{\bot\top^n \xrightarrow{a,b} \bot\top^{n+1} \mid n \geqslant 0\} \ \cup \ \{\bot\top^n \xrightarrow{c} \bot\top^{n-1} \mid n \geqslant 1\} \\ \cup \ \{\bot \xrightarrow{c} \bot\} \ \cup \ \{(\iota, \bot)\} \ \cup \ \{o\} \times \{\bot\top^n \mid n \geqslant 0\}$$

L'automate T_2 est longueur-réductible dans $\mathrm{Vis}(\{c\}, \varnothing, \{a,b\})$. Voir la figure 6.1.

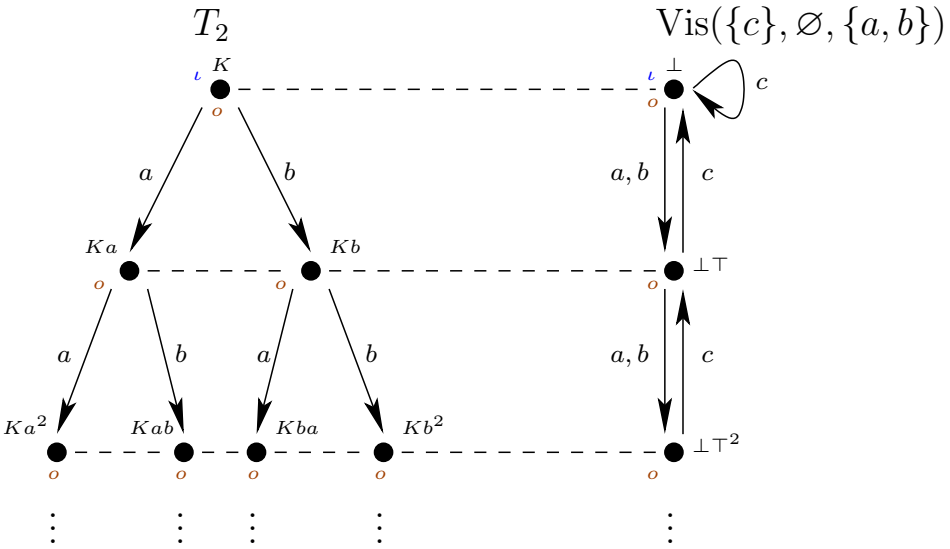


FIGURE 6.1 – T_2 est longueur-réductible dans $\mathrm{Vis}(\{c\}, \varnothing, \{a,b\})$ (Exemple 6.2.2).

La résiduation et la multiplication à droite des langages de traces par une trace préserve la régularité par niveaux.

Lemme 6.2.3. *Soit $\mathcal{L} \in \mathrm{LevelReg}(M(\Sigma, D))$ et $t \in M(\Sigma, D)$. Les langages de traces $\mathcal{L}t^{-1}$ et $\mathcal{L}t$ sont réguliers par niveaux.*

Démonstration. Pour la première assertion, observons que pour tout $[a_1 \dots a_n] \in M(\Sigma, D)$, on a

$$\mathcal{L}[a_1 \dots a_n]^{-1} = \mathcal{L}[a_n]^{-1} \dots [a_1]^{-1} \quad (n \geqslant 0)$$

Donc, il suffit de prouver que pour tout $a \in \Sigma$, $\mathcal{L}[a]^{-1}$ reste régulier par niveaux. Soit $\mathcal{A}$ un automate acceptant $L(\mathcal{A}) = \lceil \mathcal{L} \rceil_{\mathbf{F}}$. L'automate suivant accepte $\lceil \mathcal{L}[a]^{-1} \rceil_{\mathbf{F}}$.

— $(p, \bot) \xrightarrow{A} (q, \bot) : p \xrightarrow[\mathcal{A}]{A} q$

— $(p, \bot) \xrightarrow{A} (q, \top) : p \xrightarrow[\mathcal{A}]{A \dot\cup \{a\}} q$ et $A \neq \varnothing$
— $(p, \top) \xrightarrow{A} (q, \top) : p \xrightarrow[\mathcal{A}]{A} q$ et $\forall x \in A,\ xIa$
— $\{(p, \bot) \mid p \in I_{\mathcal{A}}\}$ sont les sommets initiaux.
— $\{(p, \top) \mid p \in F_{\mathcal{A}}\} \ \cup\ \{(p, \bot) \mid \exists\ q \in F_{\mathcal{A}}\ p \xrightarrow[\mathcal{A}]{a} q\}$ sont les sommets finaux.

Pour la seconde assertion, il suffit également de montrer que pour tout $a \in \Sigma$, $\mathcal{L}[a]$ est régulier par niveaux. C'est une conséquence du fait que $\lceil \mathcal{L}[a] \rceil_{\mathbf{F}}$ est l'intersection de $\mathbf{F}$ (qui est régulier par niveaux d'après le lemme 4.1.4) et du language accepté par l'automate suivant :

— $(\bot, p) \xrightarrow{A} (\bot, q) : p \xrightarrow[\mathcal{A}]{A} q$
— $(\bot, p) \xrightarrow{A \dot\cup \{a\}} (\top, q) : p \xrightarrow[\mathcal{A}]{A} q$ et $\forall x \in A,\ xIa$
— $(\top, p) \xrightarrow{A} (\top, q) : p \xrightarrow[\mathcal{A}]{A} q$ et $\forall x \in A,\ xIa$
— $(\bot, p) \xrightarrow[\mathcal{A}]{a} \top : p \in F_{\mathcal{A}}$
— $\{\bot\} \times I_{\mathcal{A}}$ sont les sommets initiaux.
— $\{\top\} \times F_{\mathcal{A}} \dot\cup \{\top\}$ sont les sommets finaux.

□

Etant donné un langage de mots L, notons $\mathrm{Pref}(L) := \{u \mid \exists\ v\ uv \in L\}$ l'ensemble des préfixes des éléments de L. A partir du lemme suivant, on déduit notamment que l'ensemble des préfixes des formes normales de Foata d'un langage (de traces) régulier par niveaux est un langage (de mots) régulier.

Lemme 6.2.4. *Soit $\mathcal{L} \in \mathrm{LevelReg}(M(\Sigma, D))$. Alors*

$$\Pi_{I_D}(\mathrm{Pref}\ \lceil \mathcal{L} \rceil_{\mathbf{F}}) \in \mathrm{LevelReg}(M(\Sigma, D))$$

6.2.2 Algèbres de Boole à partir des automates suffixes de mots

Dans ce paragraphe, nous rappelons comment diverses algèbres de Boole de langages algébriques déterministes sont obtenues à partir de la famille Stack des automates suffixes de mots [12].

Etant donné un langage L de T-mots, une famille non vide de langages $\mathcal{F}$ est une *algèbre de Boole relative à L* si $L_1 \subseteq L$, $L - L_1 \in \mathcal{F}$ et $L_1 \cap L_2 \in \mathcal{F}$ pour tout $L_1, L_2 \in \mathcal{F}$. Observons que si $\mathcal{F}$ est une algèbre de Boole relative à L, alors $L \in \mathcal{F}$.

Dans [12], Caucal et Rispal montrent comment obtenir diverses algèbres de Boole de langages algébriques déterministes.

Théorème 6.2.5 ([12])**.** *Soit H un automate suffixe de mots déterministe dont le mot vide n'est pas un sommet. Alors la classe des langages $\mathrm{Rec}^{\ell}_{\mathrm{Stack}_{\mathrm{det}}}(H) = \{L(G) \mid G \in \mathrm{Stack}_{\mathrm{det}}, G \to_{\ell} H\}$ est une algèbre de Boole relative à $L(H)$.*

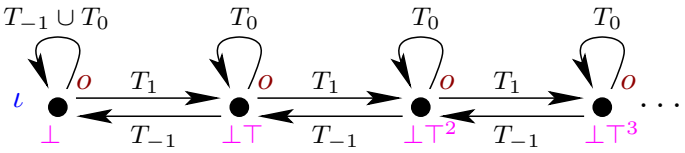


FIGURE 6.2 – L'automate à pile visible universel $\mathrm{Vis}(T_{-1}, T_0, T_1)$.

Le théorème précédent permet de retrouver l'algèbre de Boole des langages d'automates à pile visibles en terme de longueur-réduction d'automates.

Soit (T_{-1}, T_0, T_1) un alphabet visible. L'automate à pile visible universel relativement à (T_{-1}, T_0, T_1) est l'automate $\mathrm{Vis}(T_{-1}, T_0, T_1)$ défini par

Voir la figure 6.2

$$\mathrm{Vis}(T_{-1}, T_0, T_1) := \bigcup_{\lambda \in T_1} \bot\top^*(\varepsilon \xrightarrow{\lambda} \top) \,\dot\cup\, \bigcup_{\lambda \in T_0} \bot\top^*(\varepsilon \xrightarrow{\lambda} \varepsilon)$$
$$\dot\cup \bigcup_{\lambda \in T_{-1}} \bot\top^*(\top \xrightarrow{\lambda} \varepsilon) \,\dot\cup \bigcup_{\lambda \in T_{-1}} \bot(\varepsilon \xrightarrow{\lambda} \varepsilon) \,\dot\cup\, \{\iota\} \times \{\bot\} \,\dot\cup\, \{o\} \times \{\bot\top^*\}$$

Proposition 6.2.6. *Les langages (algébriques) acceptés par les automates suffixes de mots qui sont longueur-réductibles en* $\mathrm{Vis}(T_{-1}, T_0, T_1)$ *forment l'algèbre de Boole des langages d'automates à pile visibles relativement à* (T_{-1}, T_0, T_1).

6.2.3 Automates suffixes de traces avec contextes réguliers par niveaux

Nous considérons à présent la famille TrSuffix des automates suffixes de traces avec contextes réguliers par niveaux. Cette famille est une extension aux traces des automates suffixes de mots (un automate suffixe de mots est un automate suffixe de traces sur un monoïde de traces dont la relation de dépendance est totale).

Un automate suffixe de traces avec contextes réguliers par niveaux est de la forme :

$$G = \bigcup_{1 \leqslant i \leqslant n} \mathcal{W}_i(u_i \xrightarrow{a_i} v_i) \;\cup\; \{\iota\} \times \mathcal{I}_G \;\cup\; \{o\} \times \mathcal{F}_G$$

où $\mathcal{W}_i, \mathcal{I}_G, \mathcal{F}_G \in \mathrm{LevelReg}(M(\Sigma, D))$, $u_i, v_i \in M(\Sigma, D)$, $a_i \in T$ et

$$\mathcal{W}_i(u_i \xrightarrow{a_i} v_i) = \{w_i u_i \xrightarrow{a_i} w_i v_i \mid w_i \in \mathcal{W}_i\} \quad (1 \leqslant i \leqslant n)$$

. Le langage de traces $\mathcal{W}_i$ est le contexte de la règle de réécriture $\mathcal{W}_i(u_i \xrightarrow{a_i} v_i)$ $(1 \leqslant i \leqslant n)$. Notons TrSuffix la famille des automates suffixes de traces.

Exemple 6.2.7 (Arbre du quart de la grille infinie)**.** L'arbre du quart de la grille infinie est un exemple d'automate suffixe de traces. Voir la figure 5.5 et l'exemple 5.3.10.

Les automates suffixe de traces pour des relations de dépendance totales sont les automates suffixes de mots. Donc, les langages réguliers et les langages algébriques sont acceptés par les automates suffixes de traces. D'un autre côté :

Proposition 6.2.8. *Les langages acceptés par les automates suffixes de traces sont contextuels.*

Démonstration. On a montré dans le chapitre précédent qu'un automate suffixe de traces est mot-automatique. Or les automates mot-automatiques acceptent les langages contextuels [41]. □

6.2.4 Réseau de Petri généralisé

Un *réseau de Petri généralisé* G sur un alphabet Σ est un automate suffixe de traces sur $M(\Sigma, \mathrm{Id}_\Sigma)$. Notons $\mathrm{TrSuffix}^{\mathrm{Petri}}$ la famille des réseaux de Petri généralisés. La famille $\mathrm{TrSuffix}^{\mathrm{Petri}}$ est une sous-famille de TrSuffix.

Exemple 6.2.9. Soit (T_{-1}, T_0, T_1) un triplet d'alphabets finis disjoints. L'automate suffixe de traces $\mathrm{Vis}^{\mathrm{Petri}\vee\mathrm{Stack}}(T_{-1}, T_0, T_1)$ sur $M(\{\top\}, \{(\top, \top)\})$ défini ci-dessous est un réseau de Petri généralisé. Observons qu'il est longueur-isomorphe à l' automate suffixe de mots $\mathrm{Vis}(T_{-1}, T_0, T_1)$. L'algèbre de Boole $\mathrm{Rec}^{\ell}_{\mathrm{Stack}_{\mathrm{det}}}(\mathrm{Vis}^{\mathrm{Petri}\vee\mathrm{Stack}}(T_{-1}, T_0, T_1))$ (voir le théorème 6.2.5) est la famille des langages d'automates à pile visibles relativement à (T_{-1}, T_0, T_1) ([1]).

$$\mathrm{Vis}^{\mathrm{Petri}\vee\mathrm{Stack}}(T_{-1}, T_0, T_1) := \bigcup_{\lambda\in T_1} [\top^+]([\varepsilon] \xrightarrow{\lambda} [\top]) \,\dot\cup \bigcup_{\lambda\in T_0} [\top^+]([\varepsilon] \xrightarrow{\lambda} [\varepsilon])$$
$$\dot\cup \bigcup_{\lambda\in T_{-1}} [\top^+]([\top] \xrightarrow{\lambda} [\varepsilon]) \,\dot\cup \bigcup_{\lambda\in T_{-1}} [\top]([\varepsilon] \xrightarrow{\lambda} [\varepsilon]) \,\dot\cup\, \{\iota\} \times \{[\top]\} \,\dot\cup\, \{o\} \times [\top^+]$$

Dans l'exemple précédent, la variation de longueur entre deux sommets adjacents est au plus 1. Ce n'est plus le cas dans l'exemple suivant qui montre, en particulier, comment associer à un réseau de Petri au sens usuel, un réseau de Petri généralisé acceptant le même langage .

Exemple 6.2.10. Considérons le réseau de Petri suivant

$$A := \{(\binom{0}{1}, \binom{2}{0}), (\binom{0}{0}, \binom{1}{1}), (\binom{0}{0}, \binom{3}{0})\}$$

ainsi que l'automate G_A sur $\mathbb{N}^2$, défini par

$$G_A := \{v \xrightarrow{l((a_1,a_2))} v - a_1 + a_2 \mid v \in \mathbb{N}^2, a_1 \leqslant v, (a_1, a_2) \in A\}$$

où $l : A \longrightarrow T$ est la fonction d'étiquetage qui associe le terminal b à $(\binom{0}{0}, \binom{1}{1})$ et $(\binom{0}{0}, \binom{3}{0})$ et le terminal a à $(\binom{0}{1}, \binom{2}{0})$. L'automate G_A est isomorphe au réseau de Petri généralisé sur l'alphabet $\{x, y\}$ défini par les règles suivantes

$$M([y] \xrightarrow{a} [xx]) \;\cup\; M([\varepsilon] \xrightarrow{b} [xy]) \;\cup\; M([\varepsilon] \xrightarrow{b} [xxx])$$

où M désigne le monoïde de traces $M(\{x, y\}, \mathrm{Id}_{\{x,y\}})$. Voir la figure 6.3.

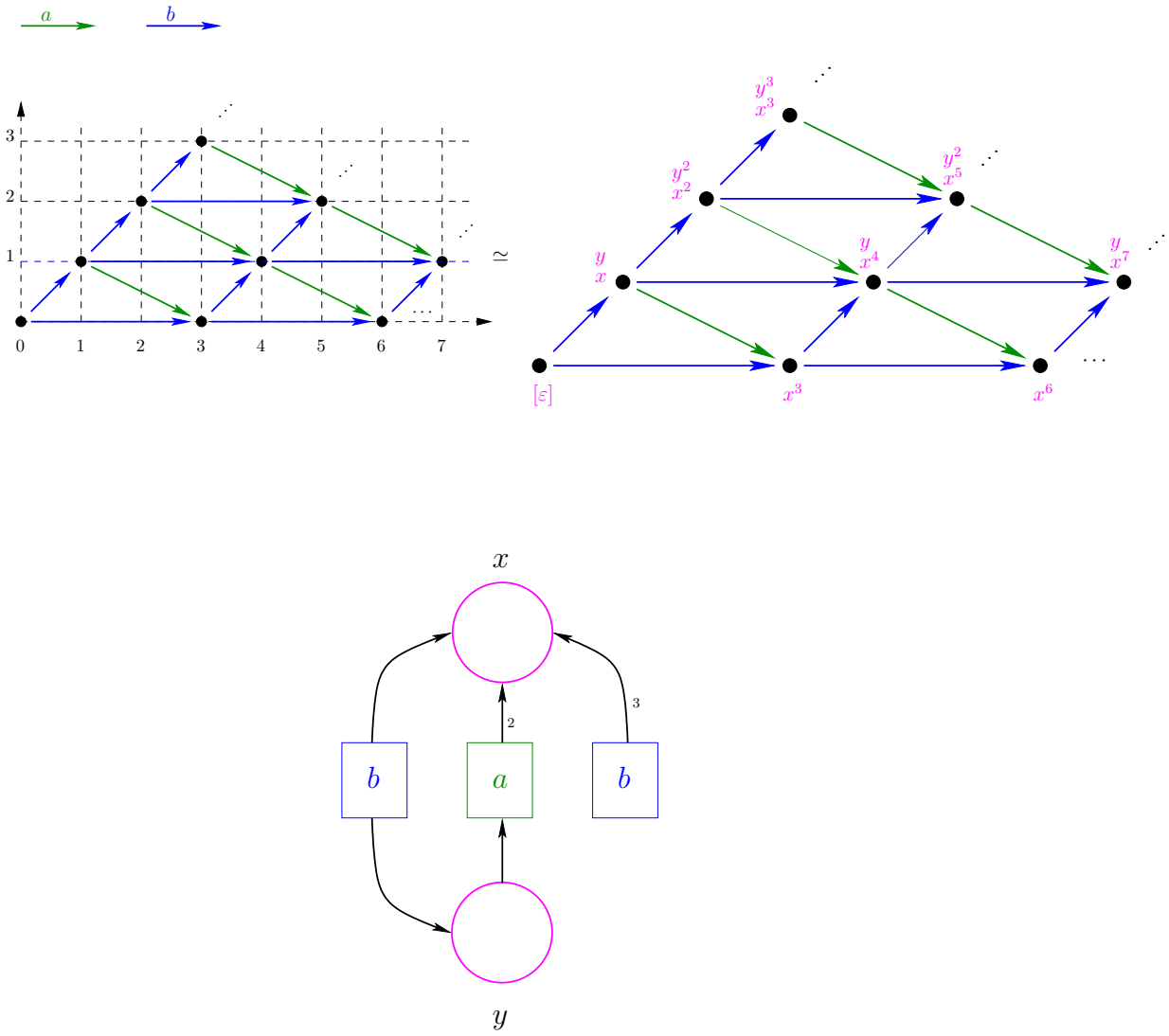


FIGURE 6.3 – Diverses représentations d'un même réseau de Petri généralisé (Exemple 6.2.10).

6.3 Synchronisation et algèbres de Boole

Dans ce paragraphe, nous montrons comment obtenir diverses algèbres de Boole de langages de mots à partir de la famille TrSuffix (théorème 6.3.24). En fait, cela sera déduit d'un résultat plus général. Plus précisément, nous définissons la synchronisation par niveaux et la superposition par niveaux de deux automates de traces (définitions 6.3.5 et 6.3.9), et nous montrons (voir le théorème 6.3.17) que si une famille $\mathcal{F}$ d'automates de traces est close par ces deux opérations, alors pour tout automate déterministe $H \in \mathcal{F}$, la famille des langages acceptés par les automates $G \in \mathcal{F}$ longueur-réductibles dans H, forment une algèbre de Boole relative au langage accepté par H. En effet, nous montrons préalablement que sous certaines conditions (lemmes 6.3.7 et 6.3.16), la synchronisation (respectivement la superposition) par niveaux de deux automates G et H (respectivement d'un automate G sur un automate H) accepte l'intersection $L(G) \cap L(H)$ (respectivement la différence $L(H) - L(G)$). Nous montrerons enfin que la famille TrSuffix est fermée par synchronisation et superposition par niveaux.

Soit $M(\Sigma_1, D_1)$ et $M(\Sigma_2, D_2)$ des monoïdes de traces tels que $\Sigma_1 \cap \Sigma_2 = \varnothing$.

Soit $s \in M(\Sigma_1, D_1)$ et $t \in M(\Sigma_2, D_2)$.

Définition 6.3.1. La synchronisation $s \parallel t$ de s et t est le produit st dans le monoïde de traces $M(\Sigma_1 \dot\cup \Sigma_2, D_1 \cup D_2)$.

Définition 6.3.2. La synchronisation par niveaux $s \parallel_= t$ de s et t est $s \parallel t$ si $\|s\| = \|t\|$ et n'est pas définie sinon.

Nous définissons aussi $s \parallel_\geqslant t := s \parallel t$ si $\|s\| \geqslant \|t\|$ et $s \parallel_\leqslant t := s \parallel t$ si $\|s\| \leqslant \|t\|$.

Etant donnés $\mathcal{L}_1 \subseteq M(\Sigma_1, D_1)$ et $\mathcal{L}_2 \subseteq M(\Sigma_2, D_2)$, nous définissons

$$\mathcal{L}_1 \parallel \mathcal{L}_2 := \{t_1 \parallel t_2 \mid t_1 \in \mathcal{L}_1, t_2 \in \mathcal{L}_2\}$$

$$\mathcal{L}_1 \parallel_= \mathcal{L}_2 := \{t_1 \parallel_= t_2 \mid t_1 \in \mathcal{L}_1, t_2 \in \mathcal{L}_2\}$$

$$\mathcal{L}_1 \parallel_\geqslant \mathcal{L}_2 := \{t_1 \parallel_\geqslant t_2 \mid t_1 \in \mathcal{L}_1, t_2 \in \mathcal{L}_2\}$$

$$\mathcal{L}_1 \parallel_\leqslant \mathcal{L}_2 := \{t_1 \parallel_\leqslant t_2 \mid t_1 \in \mathcal{L}_1, t_2 \in \mathcal{L}_2\}$$

Ces langages de traces synchronisées restent réguliers par niveaux si $\mathcal{L}_1$ et $\mathcal{L}_2$ le sont.

Lemme 6.3.3. *Soit $\mathcal{L}_1 \in \mathrm{LevelReg}(M(\Sigma_1, D_1))$ et $\mathcal{L}_2 \in \mathrm{LevelReg}(M(\Sigma_2, D_2))$. Les langages de traces suivants sont réguliers par niveaux : $\mathcal{L}_1 \parallel \mathcal{L}_2$, $\mathcal{L}_1 \parallel_\geqslant \mathcal{L}_2$, $\mathcal{L}_1 \parallel_\leqslant \mathcal{L}_2$, $\mathcal{L}_1 \parallel_= \mathcal{L}_2$.*

Démonstration. Soit $\mathcal{A}_i$ un automate tel que $L(\mathcal{A}_i) = \lceil \mathcal{L}_i \rceil_{\mathbf{F}}$ ($i \in \{1, 2\}$). Le langage accepté par l'automate ci-dessous est $\mathcal{L}_1 \parallel \mathcal{L}_2$. Si l'on retire les transitions des lignes 3, 4, 5 et 6, le langage accepté est alors $\mathcal{L}_1 \parallel_= \mathcal{L}_2$.
Si l'on retire les transitions des lignes 5 et 6 (respectivement des lignes 3 et 4), le langage accepté est alors $\mathcal{L}_1 \parallel_\geqslant \mathcal{L}_2$(respectivement $\mathcal{L}_1 \parallel_\leqslant \mathcal{L}_2$).

1. $(p_1, p_2) \xrightarrow{A_1 \dot\cup A_2} (q_1, q_2) : p_1 \xrightarrow[\mathcal{A}_1]{A_1} q_1$ et $p_2 \xrightarrow[\mathcal{A}_2]{A_2} q_2$
2. $(p_1, p_2) \xrightarrow{A_1 \dot\cup A_2} (\bot, \bot) : p_1 \xrightarrow[\mathcal{A}_1]{A_1} q_1$ et $p_2 \xrightarrow[\mathcal{A}_2]{A_2} q_2$ et $q_1 \in F_{\mathcal{A}_1}$ et $q_2 \in F_{\mathcal{A}_2}$
3. $(p_1, p_2) \xrightarrow{A_1} (q_1, \bot) : p_1 \xrightarrow[\mathcal{A}_1]{A_1} q_1$ et $p_2 \in F_{\mathcal{A}_2} \cup \{\bot\}$
4. $(p_1, \bot) \xrightarrow{A_1} (\bot, \bot) : p_1 \xrightarrow[\mathcal{A}_1]{A_1} q_1$ et $q_1 \in F_{\mathcal{A}_1} \cup \{\bot\}$
5. $(p_1, p_2) \xrightarrow{A_2} (\bot, q_2) : p_1 \in F_{\mathcal{A}_1} \cup \{\bot\}$ et $p_2 \xrightarrow[\mathcal{A}_2]{A_2} q_2$
6. $(\bot, p_2) \xrightarrow{A_2} (\bot, \bot) : p_2 \xrightarrow[\mathcal{A}_2]{A_2} q_2$ et $q_2 \in F_{\mathcal{A}_2}$
7. $I_{\mathcal{A}_1} \times I_{\mathcal{A}_2}$ est l'ensemble des sommets initiaux
8. $(\bot, \bot)$ est l'unique sommet final.

□

Si un langage de traces synchronisées est régulier par niveaux, alors ses projections le sont également. Explicitons ce qu'est une telle projection. Pour $i \in \{1, 2\}$, notons $\bar{i} := 3 - i$ et considérons le morphisme π_i de $M(\Sigma_1 \dot\cup \Sigma_2, D_1 \dot\cup D_2)$ dans $M(\Sigma_i, D_i)$ défini par $\pi_i([a]) = [a]$ si $a \in \Sigma_i$ et $\pi_i([a]) = [\varepsilon]$ si $a \in \Sigma_{\bar{i}}$. Etant donné $t \in M(\Sigma_1 \dot\cup \Sigma_2, D_1 \dot\cup D_2)$, il existe un unique couple $(t_1, t_2) \in M(\Sigma_1, D_1) \times M(\Sigma_2, D_2)$ tel que $t = t_1 t_2$. Ce couple est donné par $t_1 = \pi_1(t)$ et $t_2 = \pi_2(t)$.

Lemme 6.3.4. *Soit* $\mathcal{L} \in \mathrm{LevelReg}(M(\Sigma_1 \dot\cup \Sigma_2, D_1 \dot\cup D_2))$. *Alors*

$$\pi_i(\mathcal{L}) \in \mathrm{LevelReg}(M(\Sigma_i, D_i)) \quad (i \in \{1, 2\})$$

Démonstration. Soit $\mathcal{A}$ un automate tel que $L(\mathcal{A}) = \lceil \mathcal{L} \rceil_{\mathbf{F}}$. L'automate suivant accepte $\lceil \pi_i(\mathcal{L}) \rceil_{\mathbf{F}}$.

— $p \xrightarrow{A} q : p \xrightarrow[\mathcal{A}]{A \dot\cup B} q$, $A \in I_{D_i}$, $B \in I_{D_{\bar{i}}}$
— $I_{\mathcal{A}}$ est l'ensemble des sommets initiaux.
— $F_{\mathcal{A}}$ est l'ensemble des sommets finaux.

□

A présent, introduisons la synchronisation par niveaux de deux automates. Il s'agit d'un automate dont les sommets sont les synchronisations par niveaux des sommets de ces automates.

Définition 6.3.5. Soient G_1 et G_2 des automates de traces respectivement sur $M(\Sigma_1, D_1)$ et $M(\Sigma_2, D_2)$. Leur synchronisation par niveaux $G_1 \parallel_= G_2$ est l'automate de traces sur $M(\Sigma_1 \dot\cup \Sigma_2, D_1 \dot\cup D_2)$ défini par

$$G_1 \parallel_= G_2 := \{p_1 \parallel_= p_2 \xrightarrow{a} q_1 \parallel_= q_2 \mid p_1 \xrightarrow[G_1]{a} q_1, p_2 \xrightarrow[G_2]{a} q_2\} \cup \{\iota\} \times (I_{G_1} \parallel_= I_{G_2}) \cup \{o\} \times (F_{G_1} \parallel_= F_{G_2})$$

Le lemme suivant montre que $G_1 \parallel_= G_2$ est π_i-longueur-réductible vers G_i.

Lemme 6.3.6. *$G_1 \parallel_= G_2 \xrightarrow{\pi_i}_\ell G_i$ $(i \in \{1,2\})$.*

Démonstration. Soit $i \in \{1,2\}$ et $p_1 \parallel_= p_2 \xrightarrow{a} q_1 \parallel_= q_2 \in G_1 \parallel_= G_2$. Alors $\pi_i(p_1 \parallel_= p_2) = p_i \xrightarrow[G_i]{a} q_i = \pi_i(q_1 \parallel_= q_2)$. De manière similaire, si $(c, p_1 \parallel_= p_2) \in G_1 \parallel_= G_2$ $(c \in \{\iota, o\})$, alors $(c, \pi_i(p_1 \parallel_= p_2)) = (c, p_i) \in G_i$. Et $\|\pi_i(p_1 \parallel_= p_2)\| = \|p_i\| = \|p_1 \parallel_= p_2\|$. □

Si deux automates sont longueur-réductibles en un même automate déterministe, alors l'intersection de leurs langages acceptés est accepté par leur synchronisation par niveaux.

Lemme 6.3.7. *Si H est un automate de traces déterministe tel que $G_1 \to_\ell H$ et $G_2 \to_\ell H$, alors $L(G_1 \parallel_= G_2) = L(G_1) \cap L(G_2)$.*

Démonstration. Soit $u \in L(G_1) \cap L(G_2)$ un mot. D'après le lemme 6.2.1, puisque $G_1 \to_\ell H$, il existe un chemin acceptant étiqueté par u dans H. Puisque H est déterministe, ce chemin est unique. Alors, pour deux chemins acceptant u dans G_1 et G_2, les suites de longueur des sommets de ces chemins sont identiques car elles sont identiques à la suite des longueurs du chemin acceptant u dans H. Par conséquent, u étiquette un chemin acceptant dans $G_1 \parallel_= G_2$. L'autre inclusion est immédiate. □

Si deux automates déterministes sont longueur-réductibles en un même troisième, alors leur synchronisation par niveaux est aussi déterministe.

Lemme 6.3.8. *Si G_1, G_2 et H sont des automates de traces déterministes tels que $G_i \to_\ell H$ $(i \in \{1,2\})$, alors $G_1 \parallel_= G_2$ est un automate déterministe.*

Démonstration. Notons i_{G_i} (respectivement i_H) l'unique sommet initial de G_i (respectivement H) $(i \in \{1,2\})$. Puisque $G_i \to_\ell H$ $(i \in \{1,2\})$, nous avons $\|i_{G_1}\| = \|i_{G_2}\|$. Donc l'unique sommet initial de $G_1 \parallel_= G_2$ est $i_{G_1} \parallel_= i_{G_2}$.

De plus, puisque G_1 et G_2 sont déterministes, il ne peut pas y avoir dans $G_1 \parallel_= G_2$ deux arcs distincts de même source étiquetés par la même lettre. □

Nous allons maintenant définir l'opération de superposition par niveaux $G /\!\!/^\sharp H$ d'automates de traces. Appliquée à des automates déterministes, et si $G \to_\ell H$, on montrera que la superposition par niveaux $G /\!\!/^\sharp H$ reste longueur-réductible en H et accepte le langage différence $L(H) - L(G)$ (lemme 6.3.16).

La superposition par niveaux considérée ici a l'avantage de préserver le degré borné. En effet, nous considérerons des familles d'automates de degré borné et pour y dégager des algèbres de Boole, on demandera à ces familles d'être closes par superposition par niveaux.

Etant donné un langage de mots L et un mot u, on écrit $u \sqsubseteq L$ si u est un préfixe d'un mot dans L. Un automate de traces est $[\varepsilon]$-*libre* si $[\varepsilon]$ n'est pas un sommet.

Définition 6.3.9. Soient G et H des automates de traces $[\varepsilon]$-libres sur respectivement $M(\Sigma_1, D_1)$ et $M(\Sigma_2, D_2)$ et $\sharp \notin \Sigma_1 \dot\cup \Sigma_2$. La superposition par niveaux de G sur H est l'automate de traces $G /\!\!/^\sharp H$ sur $M(\Sigma_1 \dot\cup \{\sharp\} \dot\cup \Sigma_2, D_1 \dot\cup \{(\sharp,\sharp)\} \dot\cup D_2)$ défini par

$G /\!\!/^\sharp H :=$

(1) $\{p \parallel_= s \xrightarrow{a} q \parallel_= t \mid p \xrightarrow[G]{a} q, s \xrightarrow[H]{a} t\}$

(2) $\cup \{\iota\} \times (I_G \parallel_= I_H)$

(3) $\cup \{o\} \times ((V_G - F_G) \parallel_= F_H)$

(4) $\cup \{p \parallel_= s \xrightarrow{a} p[\sharp] \parallel t \mid s \xrightarrow[H]{a} t, \|s\| \leqslant \|t\|, p \in V_G, \neg\exists p'(p \xrightarrow[G]{a} p' \wedge \|p'\| = \|t\|)\}$

(5) $\cup \{p \parallel_= s \xrightarrow{a} q[\sharp] \parallel_= t \mid s \xrightarrow[H]{a} t, \|s\| > \|t\|, p \in V_G, \neg\exists p'(p \xrightarrow[G]{a} p' \wedge \|p'\| = \|t\|), \lceil q \rceil_{\mathbf{F}} \sqsubseteq \lceil p \rceil_{\mathbf{F}}\}$

(6) $\cup \{p[\sharp] \parallel_\leqslant s \xrightarrow{a} p[\sharp] \parallel t \mid s \xrightarrow[H]{a} t, \|s\| \leqslant \|t\|, \lceil p \rceil_{\mathbf{F}} \sqsubseteq \lceil V_G \rceil_{\mathbf{F}}\}$

(7) $\cup \{p[\sharp] \parallel_\leqslant s \xrightarrow{a} p[\sharp] \parallel_\leqslant t \mid s \xrightarrow[H]{a} t, \|s\| > \|t\|, \lceil p \rceil_{\mathbf{F}} \sqsubseteq \lceil V_G \rceil_{\mathbf{F}}\}$

(8) $\cup \{p[\sharp] \parallel_\leqslant s \xrightarrow{a} q[\sharp] \parallel_= t \mid s \xrightarrow[H]{a} t, \|s\| > \|t\|, \|t\| < \|p[\sharp]\|, \lceil p \rceil_{\mathbf{F}} \sqsubseteq \lceil V_G \rceil_{\mathbf{F}}, \lceil q \rceil_{\mathbf{F}} \sqsubseteq \lceil p \rceil_{\mathbf{F}}\}$

(9) $\cup \{o\} \times (\{p[\sharp] \parallel_\leqslant s \mid \lceil p \rceil_{\mathbf{F}} \sqsubseteq \lceil V_G \rceil_{\mathbf{F}}, s \in F_H\})$

(10) $\cup \{\iota\} \times (\{[\sharp] \parallel s \mid s \in I_H, \neg(\exists p \in I_G \ \|p\| = \|s\|)\})$

Donnons quelques explications à propos de cette définition. D'abord, observons qu'il y a deux types de sommets : ceux pour lesquels la lettre $\sharp$ apparaît et les autres. Ensuite, observons que les arcs entre les sommets ne comportant pas de $\sharp$ sont ceux de $G \parallel_= H$. Notons que les arcs aux lignes (4) et (5) sont les seuls entre sommets comportant la lettre $\sharp$ et ceux n'en comportant pas et que ces arcs quittent l'ensemble $V_{G\parallel_= H}$ des sommets ne comportant pas de $\sharp$. Par conséquent, une fois qu'un chemin quitte $V_{G\parallel_= H}$, il n'y retourne jamais.

Etant donné $t \in M(\Sigma_1 \dot\cup \{\sharp\} \dot\cup \Sigma_2, D_1 \dot\cup D_2)$, on note par $\pi_1(t)$ (respectivement $\pi_2(t)$) l'unique trace dans $M(\Sigma_1 \dot\cup \{\sharp\}, D_1)$ (respectivement $M(\Sigma_2, D_2)$) telle que $t = \pi_1(t)\pi_2(t)$.

Pour chaque arc $s \xrightarrow[H]{a} t$ et chaque sommet x de $G /\!\!/^\sharp H$, se projetant sur H en s (*i.e.*, $\pi_2(x) = s$), il existe un arc $x \xrightarrow[G/\!\!/^\sharp H]{a} y$ avec y se projetant en H sur t (*i.e.*, $\pi_2(y) = t$). Puisque d'autre part, tout sommet initial de H est relevé en un sommet initial de $G /\!\!/^\sharp H$ (ligne (2) et (10)), on en déduit que tout chemin initial dans H (un *chemin initial* est un chemin dont la source est un sommet initial) se relève en un chemin initial dans $G /\!\!/^\sharp H$.

Lemme 6.3.10. *Tout mot qui étiquette un chemin initial dans H étiquette aussi un chemin initial dans $G /\!\!/^\sharp H$.*

Enfin, notons que l'on préserve le fait que la longueur par niveaux de tout sommet est donnée par celle de sa composante dans H (voir lemme 6.3.12). En particulier, la longueur par niveaux de la première composante d'un sommet n'est pas plus grande que celle de sa deuxième composante.

Il ne sera pertinent de donner des précisions à propos des sommets finaux de $G /\!\!/^\sharp H$ que sous des hypothèses supplémentaires sur G et H(lemme 6.3.12, corollaire 6.3.14 et lemme 6.3.16).

Exemple 6.3.11. Considérons l'automate décrit dans l'exemple 6.2.2. L'unique sommet initial de $T_2 /\!\!/^\sharp \mathrm{Vis}(\{c\}, \varnothing, \{a, b\})$ est $K \parallel_= \bot$. Les sommets finaux de $T_2 /\!\!/^\sharp \mathrm{Vis}(\{c\}, \varnothing, \{a, b\})$ sont ceux ayant un $\sharp$. Voir la figure 6.4 pour la restriction de $T_2 /\!\!/^\sharp \mathrm{Vis}(\{c\}, \varnothing, \{a, b\})$ aux sommets accessibles à partir du sommet initial et co-accessibles à partir des sommets finaux.

Lemme 6.3.12. *$G /\!\!/^\sharp H \xrightarrow{\pi_2}_\ell H$.*

Démonstration. $v \xrightarrow[G/\!\!/^\sharp H]{a} w \implies \pi_2(v) \xrightarrow[H]{a} \pi_2(w)$ et $(c, v) \in G /\!\!/^\sharp H \implies (c, \pi_2(v)) \in H$ $(c \in \{\iota, o\})$.

Pour tout sommet $v \parallel s \in V_{G/\!\!/^\sharp H}$ avec $(v, s) \in M(\Sigma_1 \dot\cup \{\sharp\}, D_1) \times M(\Sigma_2, D_2)$, nous avons $\|v\| \leqslant \|s\|$. Donc $\|v \parallel s\| = \|s\|$. On en déduit que π_2 préserve la longueur. □

D'après le lemme 6.2.1, $L(G /\!\!/^\sharp H) \subseteq L(H)$. Autrement dit, si un mot étiquette un chemin acceptant dans $G /\!\!/^\sharp H$, alors il étiquette aussi un chemin acceptant dans H. La réciproque n'est pas vraie en général comme nous le verrons dans le lemme 6.3.16.

La superposition par niveaux préserve le déterminisme.

Lemme 6.3.13. *Si G et H sont déterministes, alors $G /\!\!/^\sharp H$ est déterministe.*

Observons que l'automate décrit aux lignes (1), (2) et (3) (définition 6.3.9) est la synchronisation par niveaux de H et de l'automate obtenu à partir de G en inversant la finalité des sommets (un sommet non final dans G est déclaré final et *vice-versa*). D'après le lemme 6.3.7 et puisque $H \to_\ell H$, on en déduit le corollaire ci-dessous.

Corollaire 6.3.14. *Si G et H sont déterministes, $G \to_\ell H$ et w est un mot qui étiquette un chemin initial dans $G \parallel_= H$, alors $w \in L(G) \iff w \notin L(G /\!\!/^\sharp H)$.*

Exemple 6.3.15. Donnons un exemple de superposition par niveaux $G /\!\!/^\sharp H$, où G et H sont déterministes et $G \to_\ell H$. On considère

$$G := \mathrm{Vis}(\varnothing, \varnothing, \{a\}) \setminus \{o\} \times \{\bot \top^{2n} \mid n \geqslant 0\}$$

et H un automate longueur-isomorphe à $\mathrm{Vis}(\{b\}, \varnothing, \{a\})$, de sorte que G et H soit des automates de traces sur des monoïdes de traces disjoints. Voir la figure 6.5 pour $G /\!\!/^\sharp H$ (l'automate obtenu à partir de G en inversant la finalité a été noté $\overline{G}$).

La superposition par niveaux accepte la différence.

Lemme 6.3.16. *Si G et H sont déterministes et $G \to_\ell H$, alors $L(G /\!\!/^\sharp H) = L(H) - L(G)$.*

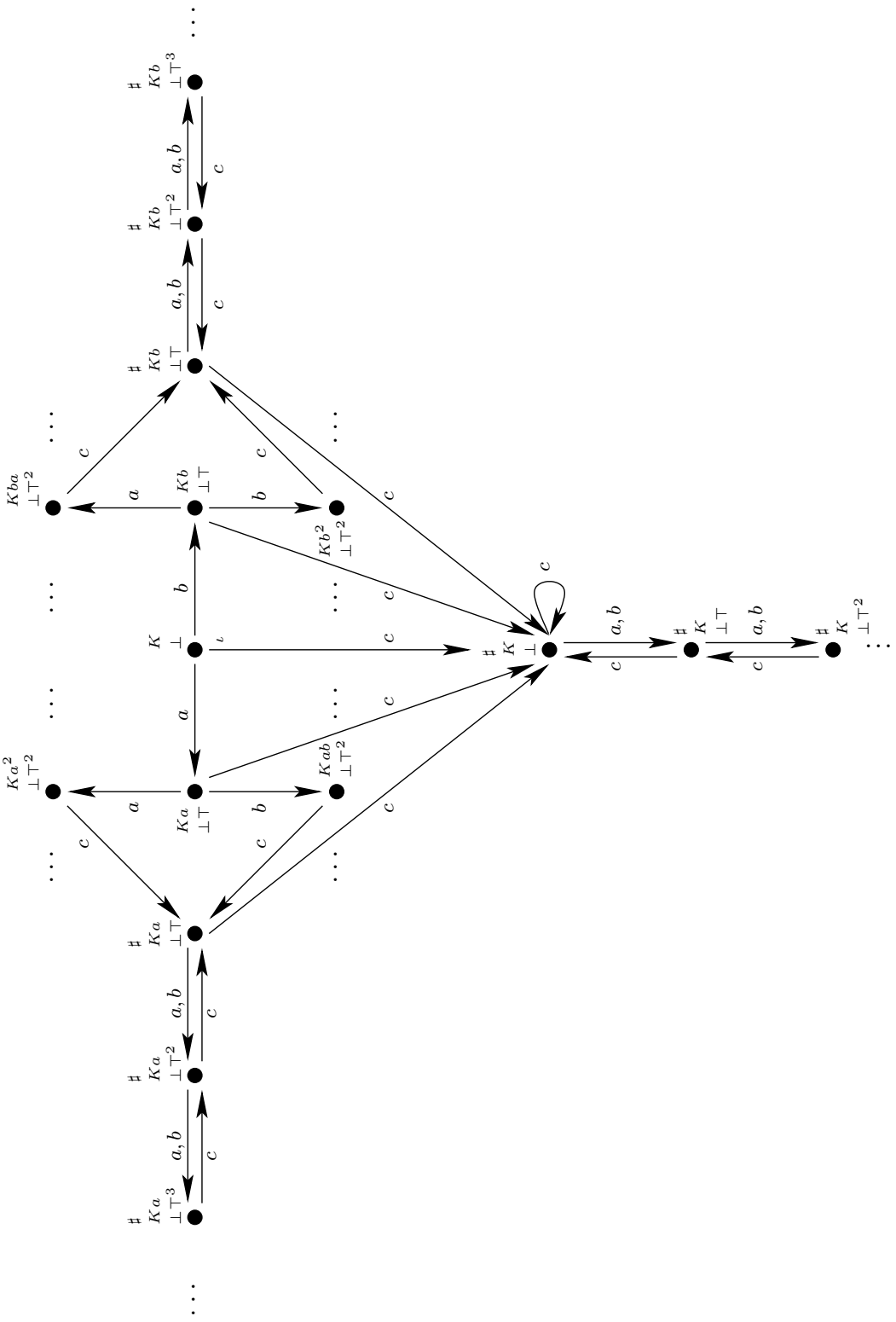


FIGURE 6.4 – Superposition par niveaux de T_2 sur $\mathrm{Vis}(\{c\}, \varnothing, \{a, b\})$ (voir l'exemple 6.3.11).

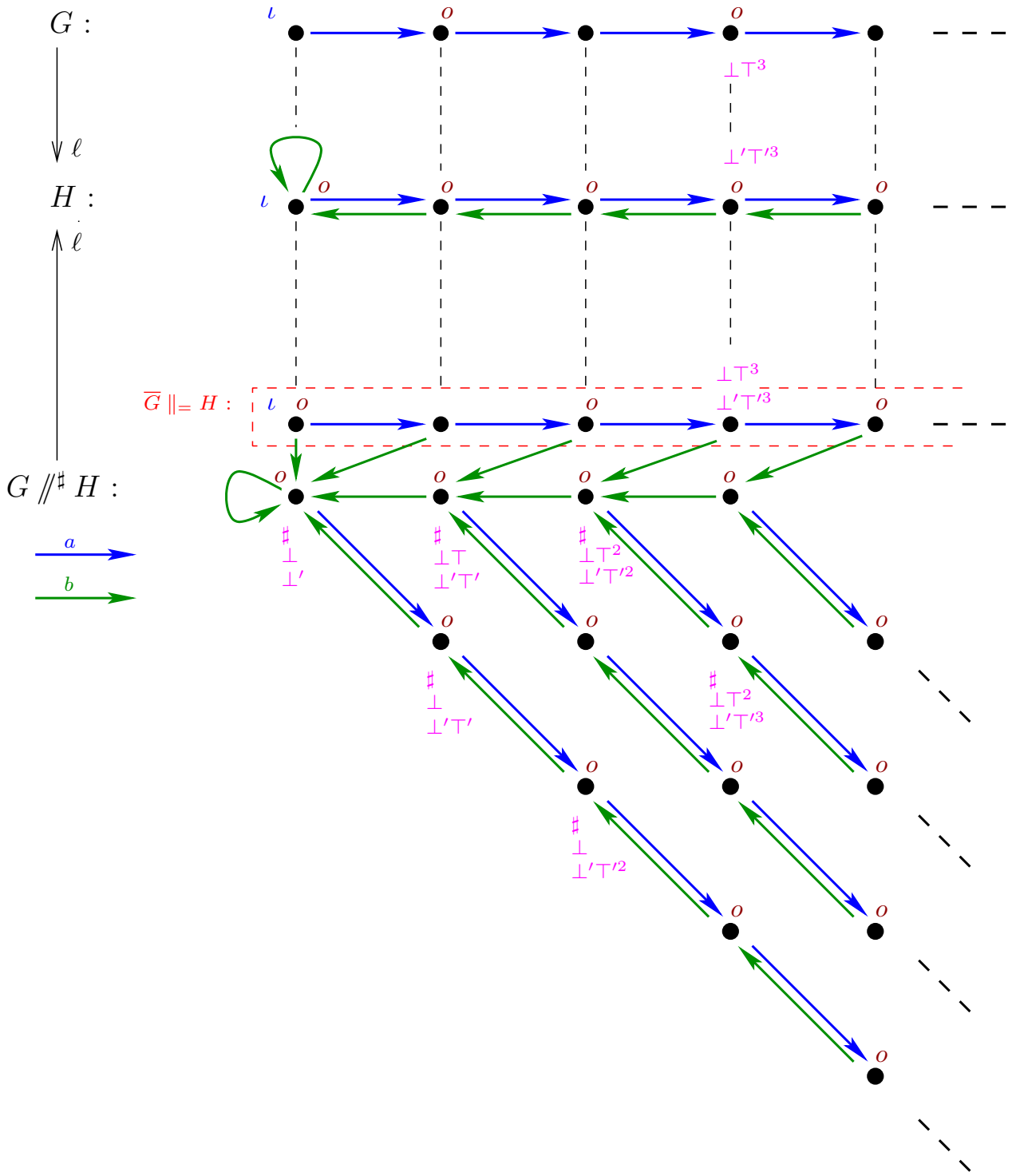


FIGURE 6.5 – Superposition par niveaux de la demi-droite sur Vis($\{b\}, \varnothing, \{a\}$) (voir l'exemple 6.3.15).

Démonstration. D'après les lemmes 6.3.12, 6.3.13 et le corollaire 6.3.14, il reste à montrer que tout mot $w \in L(H) - L(G)$ qui n'étiquette pas un chemin initial dans $G \parallel_= H$ est accepté par $G /\!\!/^\sharp H$. L'unique chemin initial dans l'automate déterministe $G /\!\!/^\sharp H$ étiqueté par w est acceptant puisque son but est de la forme $p[\sharp] \parallel_\leqslant s$ ($s \in F_H$) et qu'un tel sommet est final dans $G /\!\!/^\sharp H$ (ligne (9) dans la définition 6.3.9). □

Appliquons les lemmes 6.3.7 et 6.3.16. A partir de n'importe quelle famille d'automates de traces fermée par synchronisation par niveaux et superposition par niveaux, nous obtenons diverses algèbres de Boole de langages de mots à partir des automates déterministes de cette famille.

Théorème 6.3.17. *Soit $\mathcal{F}$ une famille d'automates de traces fermée par synchronisation par niveaux et superposition par niveaux avec $H \in \mathcal{F}_{\det}$ et $[\varepsilon]$-libre. Alors la classe de langages $\mathrm{Rec}^{\ell}_{\mathcal{F}_{\det}}(H) = \{L(G) \mid G \in \mathcal{F}_{\det}, G \to_\ell H\}$ est une algèbre de Boole relative à $L(H)$.*

Démonstration. Soient $G_1, G_2 \in \mathcal{F}_{\det}$ tels que $G_1 \to_\ell H$ et $G_2 \to_\ell H$. Nous supposons que G_1, G_2 et H sont deux à deux sur des monoïdes de trace disjoints.

Nous avons $L(H) - L(G_1) = L(G_1 /\!\!/^\sharp H)$ pour un certain symbole $\sharp$, $G_1 /\!\!/^\sharp H$ est déterministe et $G_1 /\!\!/^\sharp H \to_\ell H$. Donc $L(H) - L(G_1) \in \mathrm{Rec}^{\ell}_{\mathcal{F}_{\det}}(H)$.

De plus $L(G_1) \cap L(G_2) = L(G_1 \parallel_= G_2)$, $G_1 \parallel_= G_2$ est déterministe et $G_1 \parallel_= G_2 \to_\ell H$. Donc $L(G_1) \cap L(G_2) \in \mathrm{Rec}^{\ell}_{\mathcal{F}_{\det}}(H)$. □

Montrons que le théorème 6.3.17 s'applique à la famille TrSuffix. On commence par la fermeture par synchronisation par niveaux.

Proposition 6.3.18. *La famille* TrSuffix *est fermée par synchronisation par niveaux.*

Démonstration. D'abord, observons qu'étant donné $G_1, G_2 \in$ TrSuffix, l'ensemble des sommets initiaux de $G_1 \parallel_= G_2$ est régulier par niveaux d'après le lemme 6.3.3. En effet, la synchronisation par niveaux de langages réguliers par niveaux reste régulier par niveaux. Ensuite, puisque l'opération de synchronisation par niveaux est distributive par rapport à l'union, il suffit de supposer que G_1 (respectivement G_2) est de la forme $\mathcal{W}_1(u_1 \xrightarrow{a} v_1)$ sur $M(\Sigma_1, D_1)$ (respectivement $\mathcal{W}_2(u_2 \xrightarrow{b} v_2)$ sur $M(\Sigma_2, D_2)$) avec $\mathcal{W}_i \in \mathrm{LevelReg}(M(\Sigma_i, D_i)), u_i, v_i \in M(\Sigma_i, D_i)$ ($i \in \{1, 2\}$). Si $a \neq b$, alors $G_1 \parallel_= G_2 = \varnothing$. Supposons $a = b$. Nous allons montrer que

$$G_1 \parallel_= G_2 := \{p_1 \parallel_= p_2 \xrightarrow{a} q_1 \parallel_= q_2 \mid p_1 \xrightarrow[G_1]{a} q_1, p_2 \xrightarrow[G_2]{a} q_2\}$$

est égal à

$$G = \left((\mathcal{W}_1 u_1 \parallel_= \mathcal{W}_2 u_2)(u_1 \parallel u_2)^{-1} \cap (\mathcal{W}_1 v_1 \parallel_= \mathcal{W}_2 v_2)(v_1 \parallel v_2)^{-1}\right)(u_1 \parallel u_2 \xrightarrow{a} v_1 \parallel v_2)$$

Voir la figure 6.6.

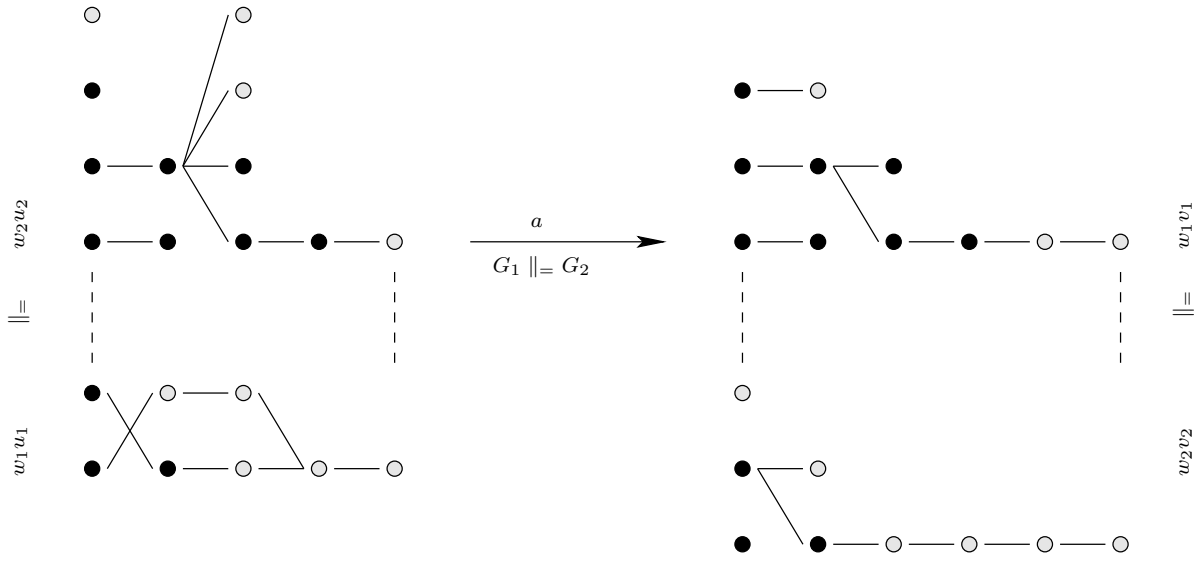


FIGURE 6.6 – Un arc dans la synchronisation par niveaux de deux automates suffixes de traces (démonstration de la proposition 6.3.18).

Puisque

$$(\mathcal{W}_1u_1 \|_= \mathcal{W}_2u_2)(u_1 \parallel u_2)^{-1} \cap (\mathcal{W}_1v_1 \|_= \mathcal{W}_2v_2)(v_1 \parallel v_2)^{-1} \in \mathrm{LevelReg}(M(\Sigma_1 \dot\cup \Sigma_2, D_1 \dot\cup D_2))$$

d'après les lemmes 6.2.3, 6.3.3 et la proposition 4.2.2, cela prouvera la proposition.

Montrons que $G_1 \|_= G_2 = G$.

($\supseteq$)

Soit $t(u_1 \parallel u_2) \xrightarrow{a} t(v_1 \parallel v_2) \in G$ avec $t \in (\mathcal{W}_1u_1 \|_= \mathcal{W}_2u_2)(u_1 \parallel u_2)^{-1} \cap (\mathcal{W}_1v_1 \|_= \mathcal{W}_2v_2)(v_1 \parallel v_2)^{-1}$. Il existe $w_1, w_1' \in \mathcal{W}_1$ et $w_2, w_2' \in \mathcal{W}_2$ tel que $t(u_1 \parallel u_2) = w_1u_1 \|_= w_2u_2 = (w_1 \parallel w_2)(u_1 \parallel u_2)$ et $t(v_1 \parallel v_2) = w_1'v_1 \|_= w_2'v_2 = (w_1' \parallel w_2')(v_1 \parallel v_2)$. Donc (un monoïde de traces est simplifiable à droite) $(w_1 \parallel w_2) = (w_1' \parallel w_2') = t$ et on déduit que $w_1 = w_1'$ et $w_2 = w_2'$. Par conséquent $t(u_1 \parallel u_2) \xrightarrow{a} t(v_1 \parallel v_2) \in G_1 \|_= G_2$.

($\subseteq$)

Soit $p_1 \|_= p_2 \xrightarrow{a} q_1 \|_= q_2 \in G_1 \|_= G_2$. Nous avons $p_1 \|_= p_2 = w_1u_1 \|_= w_2u_2 = (w_1 \parallel w_2)(u_1 \parallel u_2)$ et $q_1 \|_= q_2 = w_1v_1 \|_= w_2v_2 = (w_1 \parallel w_2)(v_1 \parallel v_2)$, avec $w_1 \in \mathcal{W}_1$ et $w_2 \in \mathcal{W}_2$. Ainsi $(w_1 \parallel w_2) \in (\mathcal{W}_1u_1 \|_= \mathcal{W}_2u_2)(u_1 \parallel u_2)^{-1} \cap (\mathcal{W}_1v_1 \|_= \mathcal{W}_2v_2)(v_1 \parallel v_2)^{-1}$ et $p_1 \|_= p_2 \xrightarrow{a} q_1 \|_= q_2 \in G$. □

Remarque 6.3.19. Soit $\mathcal{W}_i(u_i \xrightarrow{a} v_i)$ ($i \in \{1, 2\}$) un automate suffixe de traces tel que la variation par niveaux de $\mathcal{W}_i(u_i \xrightarrow{a} v_i)$ soit constante : pour chaque $i \in \{1, 2\}$, il existe un entier δ_i tel que pour tout $w_i \in \mathcal{W}_i$, $\|w_iv_i\| - \|w_iu_i\| = \delta_i$. Alors

$$\delta_1 \neq \delta_2 \implies \mathcal{W}_1(u_1 \xrightarrow{a} v_1) \|_= \mathcal{W}_2(u_2 \xrightarrow{a} v_2) = \varnothing$$

Le lemme suivant montre qu'étant donné un langage de traces régulier par niveaux $\mathcal{L}$ et un entier positif δ, l'ensemble des arcs de source $t \in \mathcal{L}$ et de but la trace obtenue à partir de t en supprimant le suffixe de longueur δ de sa forme normale de Foata, est un automate suffixe de traces.

Lemme 6.3.20. *Soit $\mathcal{L}$ un langage de traces régulier par niveaux, δ un entier positif et a un terminal. Le $\{a\}$-automate $G_{\mathcal{L}}^{\delta,a} := \{p \xrightarrow{a} q \mid p \in \mathcal{L}, \lceil q \rceil_{\mathbf{F}} \sqsubseteq \lceil p \rceil_{\mathbf{F}}, \|p\| - \|q\| = \delta\}$ est un automate suffixe de traces.*

Démonstration. L'automate $G_{\mathcal{L}}^{\delta,a}$ est égal à

$$G := \bigcup_{A_1 \dots A_\delta \in \mathbf{F}} \Pi_{I_D}(\lceil \mathcal{L} \rceil_{\mathbf{F}}(A_1 \dots A_\delta)^{-1})(\Pi_{I_D}(A_1 \dots A_\delta) \xrightarrow{a} [\varepsilon])$$

($\subseteq$) Soit $p \xrightarrow{a} q \in G_{\mathcal{L}}^{\delta,a}$. Il existe $A_1 \dots A_\delta \in \mathbf{F}$ tel que $\lceil q \rceil_{\mathbf{F}} A_1 \dots A_\delta = \lceil p \rceil_{\mathbf{F}}$. Donc $\Pi_{I_D}(\lceil q \rceil_{\mathbf{F}} A_1 \dots A_\delta) = \Pi_{I_D}(\lceil q \rceil_{\mathbf{F}})\Pi_{I_D}(A_1 \dots A_\delta) = \Pi_{I_D}(\lceil p \rceil_{\mathbf{F}}) = p$ avec $\lceil q \rceil_{\mathbf{F}} \in \lceil \mathcal{L} \rceil_{\mathbf{F}}(A_1 \dots A_\delta)^{-1}$ (puisque $\lceil q \rceil_{\mathbf{F}} A_1 \dots A_\delta = \lceil p \rceil_{\mathbf{F}} \in \lceil \mathcal{L} \rceil_{\mathbf{F}}$). Et $\Pi_{I_D}(\lceil q \rceil_{\mathbf{F}}) = q$. On en déduit que $p \xrightarrow{a} q \in G$.

($\supseteq$) Soit $A_1 \dots A_\delta \in \mathbf{F}$ et $U \in \lceil \mathcal{L} \rceil_{\mathbf{F}}(A_1 \dots A_\delta)^{-1}$. Alors

$$\Pi_{I_D}(U)\Pi_{I_D}(A_1 \dots A_\delta) = \Pi_{I_D}(UA_1 \dots A_\delta) \in \mathcal{L}$$

Et

$$\|\Pi_{I_D}(U)\Pi_{I_D}(A_1 \dots A_\delta)\| - \|\Pi_{I_D}(U)\| = \delta$$

Donc $\Pi_{I_D}(U)\Pi_{I_D}(A_1 \dots A_\delta) \xrightarrow{a} \Pi_{I_D}(U) \in G_{\mathcal{L}}^{\delta,a}$. □

Remarque 6.3.21. La conclusion du lemme 6.3.20 peut être renforcée de la façon suivante : l'automate $G_{\mathcal{L},\sharp}^{\delta,a} := \{p \xrightarrow{a} q[\sharp] \mid p \in \mathcal{L}, \lceil q \rceil_{\mathbf{F}} \sqsubseteq \lceil p \rceil_{\mathbf{F}}, \|p\| - \|q\| = \delta\}$ est un automate suffixe de traces, où $\sharp$ est une nouvelle lettre (la lettre $\sharp$ n'apparaît dans aucune trace appartenant à $\mathcal{L}$) et est indépendante de toute lettre dont elle est distincte. En fait, c'est ce dernier énoncé qui sera utilisé dans la démonstration de la proposition 6.3.23.

Dans le but de prouver la fermeture de TrSuffix par superposition par niveaux, nous montrons que toute règle de réécriture dans un automate suffixe de traces peut être décomposée en règles dont la variation niveau-longueur reste constante.

Lemme 6.3.22. *Soit H un automate suffixe de traces de la forme $\mathcal{Z}(x \xrightarrow{a} y)$. Alors il existe $\mathcal{Z}_0, \dots, \mathcal{Z}_n$ ($n \geqslant 0$) tel que $\mathcal{Z} = \dot{\bigcup}_{0 \leqslant i \leqslant n} \mathcal{Z}_i$ et pour chaque $0 \leqslant i \leqslant n$, $\|zy\| - \|zx\| = \|z'y\| - \|z'x\|$ pour tout $z, z' \in \mathcal{Z}_i$.*

Avant de prouver ce lemme, nous rappelons la définition de synchronisation lettre-à-lettre de deux mots (voir la page 60). Soit Γ un alphabet et $\sharp \notin \Gamma$ un nouveau symbole. La synchronisation de deux Γ-mots, $u = a_1 \dots a_m$ et $v = b_1 \dots b_n$, est le $(\Gamma \dot{\cup} \{\sharp\})^2$-mot $u \otimes v$ défini par

$u \otimes v := (a_1, b_1) \dots (a_k, b_k)(x_{k+1}, y_{k+1}) \dots (x_K, y_K)$, où $k = \min(m, n)$, $K = \max(m, n)$ et pour tout $k < i \leqslant K$, $(x_i, y_i) = (\sharp, b_i)$ si $k = m$ et $(x_i, y_i) = (a_i, \sharp)$ sinon.

Si un language L de $(\Gamma \dot\cup \{\sharp\})^2$-mots $u \otimes v$ est régulier, alors ses projections, à savoir les langages $\{u \in \Gamma^* \mid \exists v \in \Gamma^*\ u \otimes v \in L\}$ et $\{v \in \Gamma^* \mid \exists u \in \Gamma^*\ u \otimes v \in L\}$, sont réguliers.

Démonstration du lemme 6.3.22. D'abord montrons que l'ensemble $\{\|zy\| - \|zx\| \mid z \in \mathcal{Z}\}$ est borné.

Soit $z \in \mathcal{Z}$. D'après le lemme 5.2.14, $\|zy\| \leqslant \|z\| + |y|$ et $\|zx\| \geqslant \|z\|$. Donc $\|zy\| - \|zx\| \leqslant |y|$. De façon similaire, $-|x| \leqslant \|zy\| - \|zx\|$.

Ensuite, nous avons à vérifier que pour chaque i tel que $-|x| \leqslant i \leqslant |y|$, le langage $\mathcal{Z}_i := \{z \in \mathcal{Z} \mid \|zy\| - \|zx\| = i\}$ est régulier par niveaux.

Puisque H est mot-automatique pour le codage de l'ensemble de ses sommets par leurs formes normales de Foata, le langage $\{\lceil zx \rceil_{\mathbf{F}} \otimes \lceil zy \rceil_{\mathbf{F}} \mid z \in \mathcal{Z}\}$ est régulier. On en déduit que le langage

$$L_i := \{\lceil zx \rceil_{\mathbf{F}} \mid z \in \mathcal{Z}, \|zy\| - \|zx\| = i\}$$

est également régulier. Donc $\mathcal{Z}_i = (\Pi_{I_D}(L_i))x^{-1}$ est régulier par niveaux d'après le lemme 6.2.3. □

Proposition 6.3.23. *La famille* TrSuffix *est fermée par superposition par niveaux.*

Démonstration. Puisque $G /\!\!/^\sharp (H_1 \cup H_2) = G /\!\!/^\sharp H_1 \ \cup\ G /\!\!/^\sharp H_2$, il suffit de considérer $G = \bigcup_{1 \leqslant i \leqslant n} \mathcal{W}_i(u_i \xrightarrow{a_i} v_i) \ \cup\ \{\iota\} \times I_G \ \cup\ \{o\} \times F_G$ sur $M(\Sigma_1, D_1)$ et $H = \mathcal{Z}(x \xrightarrow{a} y) \ \cup\ \{\iota\} \times I_H \ \cup\ \{o\} \times F_H$ sur $M(\Sigma_2, D_2)$. Nous supposons que la variation par niveau des règles $\mathcal{W}_i(u_i \xrightarrow{a_i} v_i)$ $(1 \leqslant i \leqslant n)$ et $\mathcal{Z}(x \xrightarrow{a} y)$ est constante et est égale respectivement à δ_i et δ_H.

Nous montrons que les ensembles de sommets initiaux et finaux de $G /\!\!/^\sharp H$ correspondent à des langages réguliers par niveaux. Ensuite nous montrons que les ensembles d'arcs constituant $G /\!\!/^\sharp H$ (définition 6.3.9) peuvent être décrits comme des automates suffixes de traces avec contextes réguliers par niveaux. Dans chacun des cas, la régularité par niveaux des langages de traces considérés (pour les sommets initiaux, les sommets finaux et les contextes des règles de réécriture) sera assurée par la proposition 4.2.2 et les lemmes 6.2.3, 6.2.4, 6.3.3, 6.3.4.

Commençons par considérer les ensembles de sommets initiaux et finaux.

L'ensemble de sommets initiaux $I_G \parallel_= I_H$ et l'ensemble de sommets finaux $(V_G - F_G) \parallel_= F_H$ sont réguliers par niveaux.

L'ensemble de sommets finaux $\{p[\sharp] \parallel_\leqslant s \mid \lceil p \rceil_{\mathbf{F}} \sqsubseteq \lceil V_G \rceil_{\mathbf{F}}, s \in F_H\}$ et l'ensemble des sommets initiaux $\{[\sharp] \parallel s \mid s \in I_H, \neg(\exists p \in I_G\ \|p\| = \|s\|)\}$ sont égaux respectivement aux langages de traces $\Pi_{I_D}(\mathrm{Pref}(\lceil V_G \rceil_{\mathbf{F}}))[\sharp] \parallel_\leqslant F_H$ et $[\sharp] \parallel (I_H - \pi_2(I_G \parallel_= I_H))$, qui sont réguliers par niveaux.

Considérons à présent l'ensemble des arcs de $G /\!\!/^\sharp H$.

Le sous-graphe suivant de $G /\!\!/^\sharp H$

$$\{(p \parallel_= s) \xrightarrow{a} (q \parallel_= t) \mid p \xrightarrow[G]{a} q, s \xrightarrow[H]{a} t\}$$

correspond à $\bigcup_{1 \leqslant i \leqslant n} \mathcal{W}_i(u_i \xrightarrow{a_i} v_i) \parallel_= \mathcal{Z}(x \xrightarrow{a} y)$, qui est un automate suffixe de traces d'après la proposition 6.3.18.

Ensuite nous avons à identifier les sommets du synchronisé à partir desquels la synchronisation par niveaux n'est plus possible.

Affirmation 2. *Soit* $L := \{wu_i \mid 1 \leqslant i \leqslant n, a_i = a, w \in W_i, \delta_i = \delta_H\}$. *Alors* L *est régulier par niveaux et* $\{p \parallel_= zx \mid p \in V_G, z \in \mathcal{Z}, \neg\exists r(p \xrightarrow[G]{a} r \ \wedge \|r\| = \|zy\|)\} = (V_G - L) \parallel_= \mathcal{Z}x$.

Nous admettons temporairement l'affirmation précédente pour finir la démonstration. Cette affirmation sera prouvée juste après.

Nous distinguons deux cas selon le signe de δ_H et considérons les sous-graphes correspondants de $G /\!\!/^\sharp H$.

Cas 1 : $\delta_H \geqslant 0$

Les sous-graphes suivants de $G /\!\!/^\sharp H$:

$$\{p \parallel_= s \xrightarrow{a} p[\sharp] \parallel t \mid s \xrightarrow[H]{a} t, \|s\| \leqslant \|t\|, p \in V_G, \neg(\exists p'(p \xrightarrow[G]{a} p' \ \wedge \|p'\| = \|t\|))\}$$

et

$$\{p[\sharp] \parallel_\leqslant s \xrightarrow{a} p[\sharp] \parallel t \mid s \xrightarrow[H]{a} t, \|s\| \leqslant \|t\|, \lceil p \rceil_{\mathbf{F}} \sqsubseteq \lceil V_G \rceil_{\mathbf{F}}\}$$

sont égaux respectivement aux automates suffixes de traces

$$\left(((V_G - L) \parallel_= \mathcal{Z}x)\, x^{-1}\right)(x \xrightarrow{a} y[\sharp])$$

et

$$\left((\Pi_{I_D}(\mathrm{Pref}(\lceil V_G \rceil_{\mathbf{F}}))[\sharp] \parallel_\leqslant \mathcal{Z}x)\, x^{-1}\right)(x \xrightarrow{a} y)$$

Cas 2 : $\delta_H < 0$

Le sous-graphe suivant de $G /\!\!/^\sharp H$:
$\{p \parallel_= s \xrightarrow{a} q[\sharp] \parallel_= t \mid s \xrightarrow[H]{a} t, \|s\| > \|t\|, p \in V_G, \neg(\exists p'(p \xrightarrow[G]{a} p' \ \wedge \|p'\| = \|t\|)), \lceil q \rceil_{\mathbf{F}} \sqsubseteq \lceil p \rceil_{\mathbf{F}}\}$ est égal à

$$\bigcup_{A_1 \ldots A_{-\delta_H} \in \mathbf{F}} \Pi_{I_D}(\lceil V_G - L \rceil_{\mathbf{F}}(A_1 \ldots A_{-\delta_H})^{-1})(\Pi_{I_D}(A_1 \ldots A_{-\delta_H}) \xrightarrow{a} [\sharp]) \parallel_= \mathcal{Z}(x \xrightarrow{a} y)$$

qui est un automate suffixe de traces. En effet (voir la remarque 6.3.21), il s'agit de $G^{-\delta_H, a}_{V_G - L, \sharp} \parallel_= \mathcal{Z}(x \xrightarrow{a} y)$, qui est un automate suffixe de traces d'après la proposition 6.3.18.

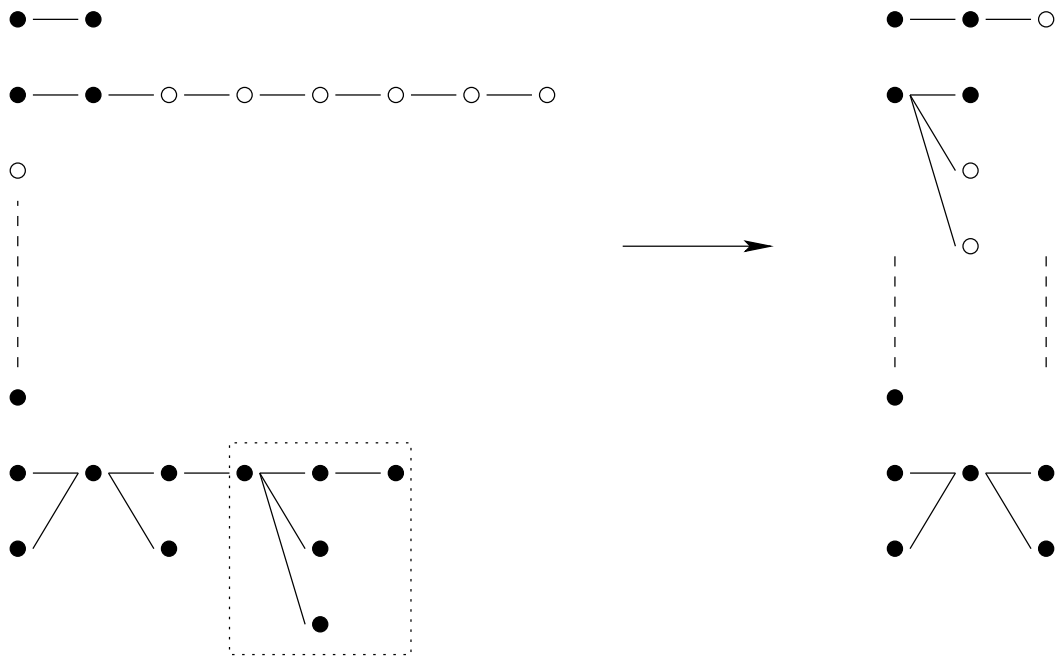


FIGURE 6.7 – Un arc de $G /\!\!/^{\sharp} H$ (démonstration de la proposition 6.3.23)

Le sous-graphe suivant de $G /\!\!/^{\sharp} H$:
$\{p[\sharp] \parallel_{\leqslant} s \xrightarrow{a} p[\sharp] \parallel_{\leqslant} t \mid s \xrightarrow[H]{a} t, \|s\| > \|t\|, \lceil p \rceil_{\mathbf{F}} \sqsubseteq \lceil V_G \rceil_{\mathbf{F}}\}$ est égal à

$$\Big((\Pi_{I_D}(\mathrm{Pref}(\lceil V_G \rceil_{\mathbf{F}}))[\sharp] \parallel_{\leqslant} \mathcal{Z}x)\, x^{-1} \ \cap\ (\Pi_{I_D}(\mathrm{Pref}(\lceil V_G \rceil_{\mathbf{F}}))[\sharp] \parallel_{\leqslant} \mathcal{Z}y)\, y^{-1}\Big)\,(x \xrightarrow{a} y)$$

Le sous-graphe suivant de $G /\!\!/^{\sharp} H$:
$\{p[\sharp] \parallel_{\leqslant} s \xrightarrow{a} q[\sharp] \parallel_{=} t \mid s \xrightarrow[H]{a} t, \|s\| > \|t\|, \|t\| < \|p[\sharp]\|, \lceil p \rceil_{\mathbf{F}} \sqsubseteq \lceil V_G \rceil_{\mathbf{F}}, \lceil q \rceil_{\mathbf{F}} \sqsubseteq \lceil p \rceil_{\mathbf{F}}\}$
est égal à

$$\bigcup_{r \in M(\Sigma_1, D_1), \|r\| \leqslant -\delta_H} (E_1 r^{-1} \ \cap\ E_2)(r \parallel x \xrightarrow{a} [\varepsilon] \parallel [y])$$

où $E_1 = (\Pi_{I_D}(\mathrm{Pref}(\lceil V_G \rceil_{\mathbf{F}}))[\sharp] \parallel_{\leqslant} \mathcal{Z}x)\, x^{-1}$ et $E_2 = (\Pi_{I_D}(\mathrm{Pref}(\lceil V_G \rceil_{\mathbf{F}}))[\sharp] \parallel_{=} \mathcal{Z}y)\, y^{-1}$.
Voir la figure 6.7.

Dans chacun des cas, il s'agit d'un automate suffixe de traces.

□

Démonstration de l'affirmation 2. Le langage de traces L est régulier par niveaux d'après la proposition 4.2.2 et le lemme 6.2.3. Par suite, il suffit de montrer que pour tout $p \in V_G$ pour lequel il existe $z \in \mathcal{Z}$ tel que $\|p\| = \|zx\|$,

$$p \in L \iff \exists p' \ (p \xrightarrow[G]{a} p' \ \wedge \|p'\| = \|zy\|)$$

($\Rightarrow$) Il existe $1 \leqslant i \leqslant n$ et $w \in \mathcal{W}_i$ tel que $a_i = a$, $p = wu_i$ et $\delta_i = \delta_H$.
Donc $p = wu_i \xrightarrow[G]{a} wv_i$ et $\|wv_i\| = \|zy\|$.

($\Leftarrow$) Il existe $1 \leqslant i \leqslant n$ tel que $a_i = a$ et $w \in \mathcal{W}_i$ tel que $p = wu_i$ et $\|wv_i\| = \|zy\|$. Par suite, $\delta_i = \|wv_i\| - \|wu_i\| = \|zy\| - \|zx\| = \delta_H$. On en déduit que $p \in L$.

□

Appliquons le théorème 6.3.17 et les propositions 6.3.18 et 6.3.23.

Théorème 6.3.24. *Soit H un automate suffixe de traces déterministe et $[\varepsilon]$-libre. Alors* $\mathrm{Rec}^{\ell}_{\mathrm{TrSuffix}_{\mathrm{det}}}(H) = \{L(G) \mid G \in \mathrm{TrSuffix}_{\mathrm{det}}, G \to_{\ell} H\}$ *est une algèbre de Boole relative à $L(H)$.*

6.4 Réseau de Petri généralisé

D'après le théorème 6.3.17, et comme nous allons prouver que $\mathrm{TrSuffix}^{\mathrm{Petri}}$ est fermée par synchronisation par niveaux et superposition par niveaux, nous obtenons diverses algèbres de Boole acceptées par des réseaux de Petri généralisés déterministes.

Théorème 6.4.1. *Soit $H \in \mathrm{TrSuffix}^{\mathrm{Petri}}_{\mathrm{det}}$ $[\varepsilon]$-libre. Alors* $\mathrm{Rec}^{\ell}_{\mathrm{TrSuffix}^{\mathrm{Petri}}_{\mathrm{det}}}(H) = \{L(G) \mid G \in \mathrm{TrSuffix}^{\mathrm{Petri}}_{\mathrm{det}}, G \to_{\ell} H\}$ *est une algèbre de Boole (relative à $L(H)$).*

Démonstration. D'après le théorème 6.3.17, il suffit de montrer que $\mathrm{TrSuffix}^{\mathrm{Petri}}$ est fermée par synchronisation par niveaux et superposition par niveaux. Considérons $G_1, G_2 \in \mathrm{TrSuffix}^{\mathrm{Petri}}$ avec G_1 sur Σ_1, G_2 sur Σ_2, $\Sigma_1 \cap \Sigma_2 = \varnothing$ et $\sharp \notin \Sigma_1 \dot\cup \Sigma_2$. Ensuite, observons que $G_1 \parallel_= G_2$ (respectivement $G_1 /\!\!/^{\sharp} G_2$) est un automate suffixe de traces sur $\Sigma_1 \dot\cup \Sigma_2$ (respectivement sur $\Sigma_1 \dot\cup \{\sharp\} \dot\cup \Sigma_2$). □

Exemple 6.4.2. L'algèbre de Boole $\mathrm{Rec}^{\ell}_{\mathrm{TrSuffix}^{\mathrm{Petri}}_{\mathrm{det}}}(\mathrm{Vis}^{\mathrm{Petri} \vee \mathrm{Stack}}(\varnothing, \varnothing, \{a\}))$ est l'algèbre de Boole des langages réguliers sur l'alphabet singleton $\{a\}$.

Exemple 6.4.3. Il est bien connu que le langage algébrique $L = \{a^n b a^n \mid n \geqslant 0\}$ n'est pas un langage accepté par un automate à pile visible [1]. Considérons le réseau de Petri généralisé G sur $\{\bot, a, b\}$ défini par

$$G := [(\bot a)^* \bot]([b] \xrightarrow{b} [\varepsilon]) \ \cup \ [b(\bot a)^* \bot]([\varepsilon] \xrightarrow{a} [\bot a]) \ \cup \ [(\bot a)^* \bot^+]([a] \xrightarrow{a} [\bot]) \cup \ \{\iota\} \times [\bot b] \ \cup \ \{o\} \times [\bot^*]$$

Le langage accepté par G vérifie

$$L = L(G) \in \mathrm{Rec}^{\ell}_{\mathrm{TrSuffix}^{\mathrm{VAS}}_{\mathrm{det}}}(\mathrm{Vis}^{\mathrm{Petri} \vee \mathrm{Stack}}(\{c\}, \{b\}, \{a\}))$$

Voir la figure 6.8.

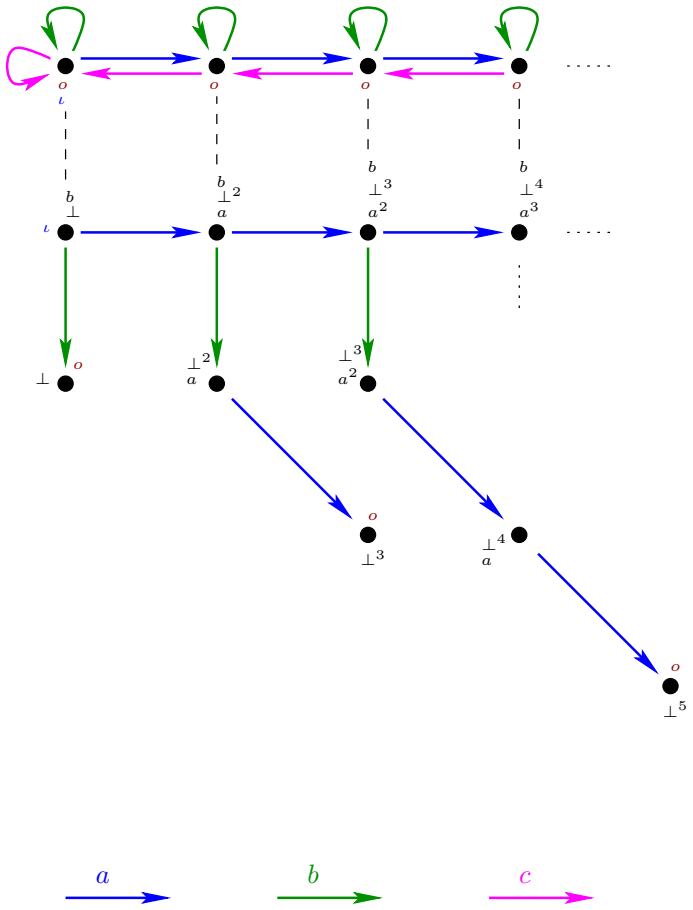


FIGURE 6.8 – Un automate suffixe de traces qui accepte $\{a^n b a^n \mid n \geqslant 0\}$ (exemple 6.4.3).

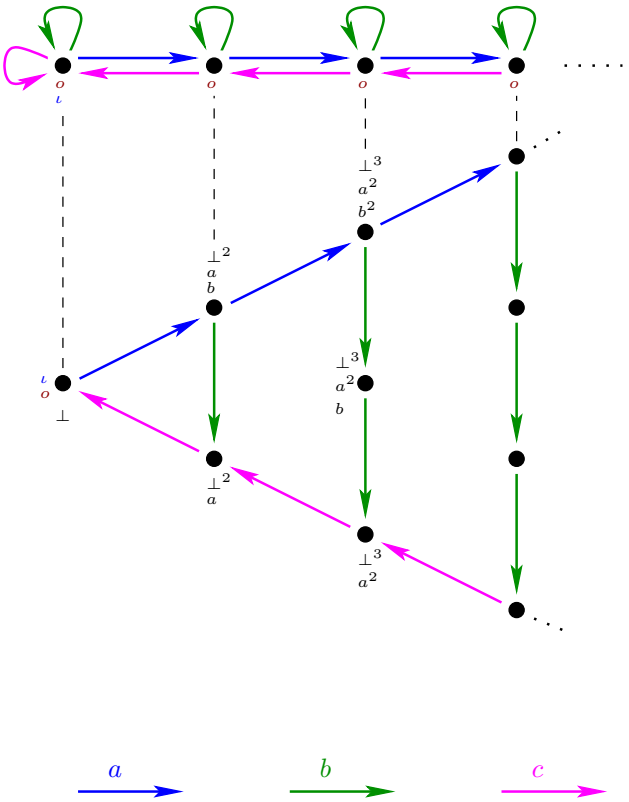


FIGURE 6.9 – Un automate suffixe de traces qui accepte $\{a^n b^n c^n \mid n \geqslant 0\}^*$ (exemple 6.4.4).

Exemple 6.4.4. Il est bien connu que le langage $L = \{a^n b^n c^n \mid n \geqslant 0\}^*$ n'est pas algébrique. Il s'agit d'un langage contextuel. Considérons le réseau de Petri généralisé G sur $\{\bot, a, b\}$ défini par

$$G := [(\bot ab)^* \bot]([\varepsilon] \xrightarrow{a} [\bot ab]) \cup [(\bot ab)^* (\bot a)^+ \bot]([b] \xrightarrow{b} [\varepsilon]) \cup [(\bot a)^* \bot]([\bot a] \xrightarrow{c} [\varepsilon]) \cup \{\iota\} \times \{[\bot]\} \cup \{o\} \times \{[\bot]\}$$

Le langage accepté par G vérifie

$$L = L(G) \in \mathrm{Rec}^{\ell}_{\mathrm{TrSuffix}^{\mathrm{VAS}}_{\mathrm{det}}}(\mathrm{Vis}^{\mathrm{Petri} \vee \mathrm{Stack}}(\{c\}, \{b\}, \{a\}))$$

Voir la figure 6.9.

Chapitre 7

Conclusion

Dans ce manuscrit, nous avons exploité la notion de traces de Mazurkiewicz afin d'appréhender la structure d'automates infinis décrivant les exécutions de systèmes concurrents. Un tel automate infini est, par exemple, le quart de la grille infinie ou encore l'arbre du quart de la grille infinie. Plus généralement, nous nous sommes intéressés aux graphes de réécriture de systèmes reconnaissables de réécriture de traces (graphes RTL). Ceux-ci ne constituent que l'extension aux traces de Mazurkiewicz des graphes de réécriture de systèmes reconnaissables de réécriture de mots dont la MSO-théorie est décidable.

Un premier travail a consisté à considérer les automates RTL d'un point de vue de la logique. Nous avons notamment montré que la théorie du premier ordre d'un automate RTL restait décidable. Nous avons également mis en évidence la sous-classe des dépliés concurrents de graphes finis concurrents :

— elle constitue plus généralement une classe de DAG mot-automatiques dont la FO[Reach]-théorie (et même la FO[Rec]-théorie) est décidable,
— il existe des dépliés concurrents de graphes finis concurrents qui ne sont pas graphes de réécriture suffixe de systèmes de réécriture de termes clos (graphes GTR), dont il est bien connu que la FO[Reach]-théorie est décidable.

Dans un deuxième travail, nous nous sommes concentrés sur les automates suffixes de traces d'un point de vue des langages qu'ils acceptent. Rappelons qu'un automate suffixe de traces est un graphe RTL dont les membres droits et gauches des règles de réécriture sont des mots. Rappelons également que nous avons appelé réseau de Petri généralisé un automate suffixe de traces sur un monoïde de traces dont la relation de dépendance est l'égalité. Nous avons montré que pour tout automate suffixe de traces déterministe (respectivement réseau de Petri généralisé déterministe) H, la classe des langages acceptés par les automates suffixes de traces déterministes (respectivement les réseaux de Petri généralisés déterministes) qui sont longueur-réductibles dans H, forment une algèbre de Boole de langages. En fait, cela est la conséquence d'un résultat plus général que nous avons montré et qui

permet de dégager diverses algèbres de Boole de langages à partir de toute famille d'automates de traces fermée par synchronisation et superposition par niveaux.

Un premier prolongement à ce travail consisterait à mieux comprendre « les systèmes concurrents de niveau 2 », c'est-à-dire ceux qui étendent le niveau 2 de la hiérarchie à pile. Plus précisément, comme les graphes de la hiérarchie à pile sont précisément les graphes de transitions des automates à pile d'ordre supérieur (au sens large), se pose donc la question d'un modèle équivalent dans le cas des traces de Mazurkiewicz. Un deuxième prolongement consisterait à expliciter diverses algèbres de Boole obtenues, ou encore d'en dégager d'éventuelles propriétés de fermeture supplémentaires.

Bibliographie

[1] R. Alur et P. Madhusudan. "Visibly pushdown languages". In : *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*. 2004, p. 202-211. doi : 10.1145/1007352.1007390. url : https://doi.org/10.1145/1007352.1007390.

[2] M. Benois. "Parties rationnelles du groupe libre". In : *C. R. Acad. Sci. Paris*. Sér. A 269 (1969), p. 1188 -1190.

[3] J. Büchi. "On a Decision Method in Restricted Second Order Arithmetic". In : *The Collected Works of J. Richard Büchi*. Sous la dir. de Saunders Mac Lane et Dirk Siefkes. New York, NY : Springer New York, 1990, p. 425-435. isbn : 978-1-4613-8928-6. doi : 10.1007/978-1-4613-8928-6_23. url : https://doi.org/10.1007/978-1-4613-8928-6_23.

[4] A. Carayol et C. Morvan. "On Rational Trees". In : *Computer Science Logic*. Sous la dir. de Zoltán Ésik. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, p. 225-239. isbn : 978-3-540-45459-5.

[5] A. Carayol et S. Wöhrle. "The Caucal Hierarchy of Infinite Graphs in Terms of Logic and Higher-Order Pushdown Automata". In : *FST TCS 2003 : Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*. 2003, p. 112-123. doi : 10.1007/978-3-540-24597-1_10.

[6] D. Caucal. "Boolean algebras of unambiguous context-free languages". In : *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2008, December 9-11, 2008, Bangalore, India*. 2008, p. 83-94. doi : 10.4230/LIPIcs.FSTTCS.2008.1743. url : https://doi.org/10.4230/LIPIcs.FSTTCS.2008.1743.

[7] D. Caucal. "Deterministic graph grammars". In : *Logic and Automata : History and Perspectives [in Honor of Wolfgang Thomas]*. 2008, p. 169-250. doi : 10.5117/9789053565766.

[8] D. Caucal. "On Infinite Terms Having a Decidable Monadic Theory". In : *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*. 2002,

p. 165-176. DOI : 10.1007/3-540-45687-2_13. URL : https://doi.org/10.1007/3-540-45687-2_13.

[9] D. CAUCAL. "On the Regular Structure of Prefix Rewriting". In : *CAAP '90, 15th Colloquium on Trees in Algebra and Programming, Copenhagen, Denmark, May 15-18, 1990, Proceedings*. 1990, p. 87-102. DOI : 10.1007/3-540-52590-4_42.

[10] D. CAUCAL. "On the Regular Structure of Prefix Rewriting". In : *Theor. Comput. Sci.* 106.1 (1992), p. 61-86. DOI : 10.1016/0304-3975(92)90278-N. URL : https://doi.org/10.1016/0304-3975(92)90278-N.

[11] D. CAUCAL. "On the transition graphs of turing machines". In : *Theor. Comput. Sci.* 296.2 (2003), p. 195-223. DOI : 10.1016/S0304-3975(02)00655-2. URL : https://doi.org/10.1016/S0304-3975(02)00655-2.

[12] D. CAUCAL et C. RISPAL. "Boolean Algebras by Length Recognizability". In : *Models, Mindsets, Meta : The What, the How, and the Why Not ? Essays Dedicated to Bernhard Steffen on the Occasion of His 60th Birthday*. Sous la dir. de Tiziana MARGARIA, Susanne GRAF et Kim G. LARSEN. Cham : Springer International Publishing, 2019, p. 169-185. ISBN : 978-3-030-22348-9. DOI : 10.1007/978-3-030-22348-9_11. URL : https://doi.org/10.1007/978-3-030-22348-9_11.

[13] N. CHOMSKY. "Three models for the description of language". In : *IRE Transactions on Information Theory* 2.3 (1956), p. 113-124. ISSN : 0096-1000. DOI : 10.1109/TIT.1956.1056813.

[14] T. COLCOMBET et C. LÖDING. "Transforming structures by set interpretations". In : *Logical Methods in Computer Science* 3.2 (2007). DOI : 10.2168/LMCS-3(2:4)2007.

[15] B. COURCELLE et I. WALUKIEWICZ. "Monadic second-order logic, graph coverings and unfoldings of transition systems". In : *Annals of Pure and Applied Logic* 92.1 (1998), p. 35 -62. ISSN : 0168-0072. DOI : https://doi.org/10.1016/S0168-0072(97)00048-1. URL : http://www.sciencedirect.com/science/article/pii/S0168007297000481.

[16] S. CRESPI-REGHIZZI et D. MANDRIOLI. "Petri nets and szilard languages". In : *Information and Control* 33.2 (1977), p. 177 -192. ISSN : 0019-9958. DOI : https://doi.org/10.1016/S0019-9958(77)90558-7. URL : http://www.sciencedirect.com/science/article/pii/S0019995877905587.

[17] M. DAUCHET et S. TISON. "The Theory of Ground Rewrite Systems is Decidable". In : *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990*. 1990, p. 242-248. DOI : 10.1109/LICS.1990.113750.

[18] V. Diekert et G. Rozenberg. *The Book of Traces*. WORLD SCIENTIFIC, 1995. DOI : 10.1142/2563. eprint : https://www.worldscientific.com/doi/pdf/10.1142/2563. URL : https://www.worldscientific.com/doi/abs/10.1142/2563.

[19] S. Eilenberg. *Automata, Languages, and Machines*. Orlando, FL, USA : Academic Press, Inc., 1974. ISBN : 0122340019.

[20] M. Hack. *DECIDABILITY QUESTIONS FOR PETRI NETS*. Rapp. tech. Cambridge, MA, USA, 1976.

[21] B. Hodgson. "On Direct Products of Automaton Decidable Theories". In : *Theor. Comput. Sci.* 19 (1982), p. 331-335. DOI : 10.1016/0304-3975(82)90042-1.

[22] M. Jantzen. "On the hierarchy of Petri net languages". en. In : *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications* 13.1 (1979), p. 19-30. URL : http://www.numdam.org/item/ITA_1979__13_1_19_0.

[23] S. Kuroda. "Classes of languages and linear-bounded automata". In : *Information and Control* 7.2 (1964), p. 207 -223. ISSN : 0019-9958. DOI : https://doi.org/10.1016/S0019-9958(64)90120-2. URL : http://www.sciencedirect.com/science/article/pii/S0019995864901202.

[24] J. Leroux, V. Penelle et G. Sutre. "On the Context-Freeness Problem for Vector Addition Systems". In : *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*. 2013, p. 43-52. DOI : 10.1109/LICS.2013.9. URL : https://doi.org/10.1109/LICS.2013.9.

[25] P. Madhusudan. "Model-checking Trace Event Structures". In : *18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22-25 June 2003, Ottawa, Canada, Proceedings*. 2003, p. 371-380. DOI : 10.1109/LICS.2003.1210077.

[26] A. Mansard. "Boolean Algebras from Trace Automata". In : *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019, December 11-13, 2019, Bombay, India*. 2019, 48 :1-48 :15. DOI : 10.4230/LIPIcs.FSTTCS.2019.48. URL : https://doi.org/10.4230/LIPIcs.FSTTCS.2019.48.

[27] A. Mansard. "Unfolding of Finite Concurrent Automata". In : *Proceedings 11th Interaction and Concurrency Experience, ICE 2018, Madrid, Spain, June 20-21, 2018*. 2018, p. 68-84. DOI : 10.4204/EPTCS.279.8. URL : https://doi.org/10.4204/EPTCS.279.8.

[28] A. Mazurkiewicz. “Trace theory”. In : *Petri Nets : Applications and Relationships to Other Models of Concurrency*. Sous la dir. de W. Brauer, W. Reisig et G. Rozenberg. Berlin, Heidelberg : Springer Berlin Heidelberg, 1987, p. 278-324. isbn : 978-3-540-47926-0.

[29] J. McKnight. “Kleene quotient theorems.” In : *Pacific J. Math.* 14.4 (1964), p. 1343-1352. url : https://projecteuclid.org:443/euclid.pjm/1103033807.

[30] K. Mehlhorn. “Pebbling Moutain Ranges and its Application of DCFL-Recognition”. In : *Automata, Languages and Programming, 7th Colloquium, Noordweijkerhout, The Netherlands, July 14-18, 1980, Proceedings*. 1980, p. 422-435. doi : 10.1007/3-540-10003-2_89. url : https://doi.org/10.1007/3-540-10003-2_89.

[31] C. Morvan. “On Rational Graphs”. In : *Foundations of Software Science and Computation Structures, Third International Conference, FOSSACS 2000, Held as Part of the Joint European Conferences on Theory and Practice of Software,ETAPS 2000, Berlin, Germany, March 25 - April 2, 2000, Proceedings*. 2000, p. 252-266. doi : 10.1007/3-540-46432-8_17. url : https://doi.org/10.1007/3-540-46432-8_17.

[32] C. Morvan et C. Stirling. “Rational Graphs Trace Context-Sensitive Languages”. In : *Mathematical Foundations of Computer Science 2001, 26th International Symposium, MFCS 2001 Marianske Lazne, Czech Republic, August 27-31, 2001, Proceedings*. 2001, p. 548-559. doi : 10.1007/3-540-44683-4_48. url : https://doi.org/10.1007/3-540-44683-4_48.

[33] D. Muller et P. Schupp. “The Theory of Ends, Pushdown Automata, and Second-Order Logic”. In : *Theor. Comput. Sci.* 37 (1985), p. 51-75. doi : 10.1016/0304-3975(85)90087-8. url : https://doi.org/10.1016/0304-3975(85)90087-8.

[34] J. Myhill. “Linear Bounded Automata”. In : *Wright Air Development Division* (1960). url : https://ci.nii.ac.jp/naid/10006925353/en/.

[35] D. Nowotka et J. Srba. “Height-Deterministic Pushdown Automata”. In : *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Ceský Krumlov, Czech Republic, August 26-31, 2007, Proceedings*. 2007, p. 125-134. doi : 10.1007/978-3-540-74456-6_13. url : https://doi.org/10.1007/978-3-540-74456-6_13.

[36] M. Parigot et E. Pelz. “A Logical Approach of Petri Net Languages”. In : *Theor. Comput. Sci.* 39 (1985), p. 155-169. doi : 10.1016/0304-3975(85)90136-7. url : https://doi.org/10.1016/0304-3975(85)90136-7.

[37] V. Penelle. "Rewriting Higher-Order Stack Trees". In : *Computer Science - Theory and Applications - 10th International Computer Science Symposium in Russia, CSR 2015, Listvyanka, Russia, July 13-17, 2015, Proceedings*. 2015, p. 364-397. DOI : 10.1007/978-3-319-20297-6_24. URL : https://doi.org/10.1007/978-3-319-20297-6_24.

[38] J. Peterson. *Petri Net Theory and the Modeling of Systems*. Upper Saddle River, NJ, USA : Prentice Hall PTR, 1981. ISBN : 0136619835.

[39] J. Peterson. *Petri Net Theory and the Modeling of Systems*. USA : Prentice Hall PTR, 1981. ISBN : 0136619835.

[40] M. Rabin. "Decidability of second-order theories and automata on infinite trees". In : *Bull. Amer. Math. Soc.* 74.5 (sept. 1968), p. 1025-1029. URL : https://projecteuclid.org:443/euclid.bams/1183529958.

[41] C. Rispal. "The synchronized graphs trace the context-sensitive languages". In : *Electr. Notes Theor. Comput. Sci.* 68.6 (2002), p. 55-70. DOI : 10.1016/S1571-0661(04)80533-4. URL : https://doi.org/10.1016/S1571-0661(04)80533-4.

[42] S. Schwer. "The context-freeness of the languages associated with vector addition systems is decidable". In : *Theoretical Computer Science* 98.2 (1992), p. 199 -247. ISSN : 0304-3975. DOI : https://doi.org/10.1016/0304-3975(92)90002-W. URL : http://www.sciencedirect.com/science/article/pii/030439759290002W.

[43] A. Semenov. "Decidability of monadic theories". In : *Mathematical Foundations of Computer Science 1984*. Sous la dir. de M. P. Chytil et V. Koubek. Berlin, Heidelberg : Springer Berlin Heidelberg, 1984, p. 162-175. ISBN : 978-3-540-38929-3.

[44] M. Shields. "Asynchronous Transition Systems". In : *Semantics of Parallelism : Non-Interleaving Representation of Behaviour*. London : Springer London, 1997, p. 183-189. ISBN : 978-1-4471-0933-4. DOI : 10.1007/978-1-4471-0933-4_15.

[45] P. Thiagarajan. "Regular Trace Event Structures". In : *BRICS Report Series* 3.32 (1996). DOI : 10.7146/brics.v3i32.20012. URL : https://tidsskrift.dk/brics/article/view/20012.

[46] W. Thomas. "A Short Introduction to Infinite Automata". In : *Developments in Language Theory, 5th International Conference, DLT 2001, Vienna, Austria, July 16-21, 2001, Revised Papers*. 2001, p. 130-144. DOI : 10.1007/3-540-46011-X_10. URL : https://doi.org/10.1007/3-540-46011-X_10.

[47] R. Valk et G. Vidal-Naquet. “Petri nets and regular languages”. In : *Journal of Computer and System Sciences* 23.3 (1981), p. 299 -325. issn : 0022-0000. doi : https://doi.org/10.1016/0022-0000(81)90067-2. url : http://www.sciencedirect.com/science/article/pii/0022000081900672.

[48] I. Walukiewicz. “Monadic second order logic on tree-like structures”. In : *STACS 96*. Sous la dir. de Claude Puech et Rüdiger Reischuk. Berlin, Heidelberg : Springer Berlin Heidelberg, 1996, p. 399-413. isbn : 978-3-540-49723-3.

[49] S. Wöhrle et W. Thomas. “Model Checking Synchronized Products of Infinite Transition Systems”. In : *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*. 2004, p. 2-11. doi : 10.1109/LICS.2004.1319595.

UNIVERSITÉ DE LA RÉUNION

Pôle Recherche
Ecoles doctorales

LETTRE D'ENGAGEMENT DE NON-PLAGIAT

Je, soussigné(e) Alexandre Mansard en ma qualité de doctorant(e) de l'Université de La Réunion, déclare être conscient(e) que le plagiat est un acte délictueux passible de sanctions disciplinaires. Aussi, dans le respect de la propriété intellectuelle et du droit d'auteur, je m'engage à systématiquement citer mes sources, quelle qu'en soit la forme (textes, images, audiovisuel, internet), dans le cadre de la rédaction de ma thèse et de toute autre production scientifique, sachant que l'établissement est susceptible de soumettre le texte de ma thèse à un logiciel anti-plagiat.

Fait à Saint-Denis le : 07/11/2020

Signature : 

Extrait du Règlement intérieur de l'Université de La Réunion
(validé par le Conseil d'Administration en date du 11 décembre 2014)

Article 9. Protection de la propriété intellectuelle – Faux et usage de faux, contrefaçon, plagiat

L'utilisation des ressources informatiques de l'Université implique le respect de ses droits de propriété intellectuelle ainsi que ceux de ses partenaires et plus généralement, de tous tiers titulaires de tes droits.
En conséquence, chaque utilisateur doit :
- utiliser les logiciels dans les conditions de licences souscrites ;
- ne pas reproduire, copier, diffuser, modifier ou utiliser des logiciels, bases de données, pages Web, textes, images, photographies ou autres créations protégées par le droit d'auteur ou un droit privatif, sans avoir obtenu préalablement l'autorisation des titulaires de ces droits.

La contrefaçon et le faux
Conformément aux dispositions du code de la propriété intellectuelle, toute représentation ou reproduction intégrale ou partielle d'une œuvre de l'esprit faite ans le consentement de son auteur est illicite et constitue un délit pénal.
L'article 444-1 du code pénal dispose : « Constitue un faux toute altération frauduleuse de la vérité, de nature à cause un préjudice et accomplie par quelque moyen que ce soit, dans un écrit ou tout autre support d'expression de la pensée qui a pour objet ou qui peut avoir pour effet d'établir la preuve d'un droit ou d'un fait ayant des conséquences juridiques ».
L'article L335_3 du code de la propriété intellectuelle précise que : « Est également un délit de contrefaçon toute reproduction, représentation ou diffusion, par quelque moyen que ce soit, d'une œuvre de l'esprit en violation des droits de l'auteur, tels qu'ils sont définis et réglementés par la loi. Est également un délit de contrefaçon la violation de l'un des droits de l'auteur d'un logiciel (…) ».

Le plagiat est constitué par la copie, totale ou partielle d'un travail réalisé par autrui, lorsque la source empruntée n'est pas citée, quel que soit le moyen utilisé. Le plagiat constitue une violation du droit d'auteur (au sens des articles L 335-2 et L 335-3 du code de la propriété intellectuelle). Il peut être assimilé à un délit de contrefaçon. C'est aussi une faute disciplinaire, susceptible d'entraîner une sanction.
Les sources et les références utilisées dans le cadre des travaux (préparations, devoirs, mémoires, thèses, rapports de stage…) doivent être clairement citées. Des citations intégrales peuvent figurer dans les documents rendus, si elles sont assorties de leur référence (nom d'auteur, publication, date, éditeur…) et identifiées comme telles par des guillemets ou des italiques.

Les délits de contrefaçon, de plagiat et d'usage de faux peuvent donner lieu à une sanction disciplinaire indépendante de la mise en œuvre de poursuites pénales.