
Prédicats algébriques d'entiers

Rapport de stage :

Master 2 de recherche en informatique

IRISA : Galion

T. Lavergne

Juin 2005

Table des matières

1 Définitions	5
1.1 Mots	5
1.2 Graphes	6
1.3 Arbres	8
1.4 Automates	10
1.4.1 Automates généraux	10
1.4.2 Automates à pile	11
1.5 Transducteurs	13
2 La hiérarchie	15
2.1 Familles de graphes infinis	15
2.1.1 Une première famille de graphes infinis	15
2.1.2 Graphes réguliers	16
2.1.3 Graphes préfixe reconnaissables	18
2.2 Théories logiques	18
2.2.1 Logique au premier ordre	19
2.2.2 Logique au second ordre monadique	20
2.3 Transformations de graphes	21
2.4 La hiérarchie	23
3 Mots infinis dans la hiérarchie	25
3.1 Mot de Champernowne	25
3.2 Mot de Liouville	26
3.3 D'autres mots infinis	27
4 Arbres algébriques et mots infinis	28
4.1 Grammaires de termes	28
4.2 Forme réduite des grammaires de termes	31
4.2.1 Grammaire de termes développée	31
4.2.2 Grammaire de terme élaguée	33
4.2.3 Grammaire de terme récursive simple	34
4.2.4 Forme réduite	35
4.3 Mots infinis au deuxième niveau	36

4.3.1 Arbres algébriques	36
4.4 Perspectives	38
5 Transducteurs et mots infinis	39
5.1 Transducteurs strictement croissants	39
5.2 Perspectives	40

Introduction

Les graphes sont des structures bien connues et très utilisées dans de nombreux domaines de l'informatique. A l'aide de transformations, il est possible d'engendrer, à partir d'une famille connue de graphes, une nouvelle famille de graphes. Au cours de ce stage de master, nous sommes intéressés à la hiérarchie de graphes définie par Caucal, à partir de deux transformations qui ont la particularité de préserver la décidabilité de la logique au second ordre monadique.

Cette hiérarchie est donc particulièrement intéressante, car des propriétés telles que l'accessibilité d'un sommet ou l'existence d'un chemin infini sont décidables.

Mais les différentes familles de graphes que l'on y trouve sont mal connues. Le niveau 0 est la famille des graphes finis, et le niveau 1 est la famille des graphes dits préfixe-reconnaissables. Ces deux familles ont déjà été étudiées en détails, mais dès le deuxième niveau, on ne connaît que peu de choses.

L'objectif de ce stage était donc d'étudier ces niveaux, et plus précisément les mots infinis que l'on peut y trouver. En effet, les mots infinis sont parmi les graphes les plus simples que l'on puisse rechercher dans ces familles, mais ils sont un premier pas vers une meilleure compréhension de la hiérarchie et donnent une base pour l'étude de graphes plus complexes.

Nous commencerons par une présentation des graphes, et de la logique au second ordre monadique, puis de la hiérarchie en elle-même, avant d'étudier quelques mots infinis particuliers. Ensuite, nous verrons de manière plus précise le niveau 2 et les mots infinis qu'il contient, que nous essayerons de caractériser. Nous terminerons sur une autre famille de graphes engendrés par des transducteurs, qui est liée à la hiérarchie, et dont nous étudierons, là aussi, les mots infinis.

Chapitre 1

Définitions

1.1 Mots

L'objectif de ce stage étant de caractériser des mots, nous allons donc commencer par les définir

Une *lettre* est un élément d'un ensemble Γ fini quelconque que l'on appellera *alphabet*.

Un *mot fini* est une suite finie quelconque de lettres dans Γ . La longueur d'un mot fini u est le nombre d'occurrences de lettres qui le compose et sera notée $|u|$. Le *mot vide* est le mot de longueur nulle, que l'on notera ε . L'ensemble de tous les mots sur Γ est noté Γ^* .

Un *mot infini* est une suite infinie de lettre; par convention sa longueur sera notée ω .

Exemple 1 *Les suites de lettres $abba$ et $aaabb$ sont des mots finis sur l'alphabet contenant les lettres a et b . Le mot $abababab\dots$ est infini sur ce même alphabet.*

Dans la suite, un mot sera toujours considéré fini sauf mention explicite du contraire.

La *concaténation* d'un mot fini u et d'un mot fini ou infini v donne le mot constitué des lettres de u suivies des lettres de v et se note $u \cdot v$ ou uv .

Exemple 2 *Le mot $abba$ s'obtient en concaténant les mots ab et ba . Le mot $abbaababab\dots$ est obtenu en concaténant le mot précédent au mot infini $abab\dots$.*

Le mot vide est l'élément neutre de la concaténation : $u \cdot \varepsilon = \varepsilon \cdot u = u$.

On notera u^n , le mot constitué par la concaténation $n \geq 0$ fois du mot fini u . On peut étendre cette notion aux puissances infinies : le mot u^ω est le mot constitué de la répétition infinie du mot u .

Soit u un mot fini ou infini, si x et y sont deux mots finis et z un mot fini ou infini tel que $u = xyz$, alors x , y et z sont appelés respectivement *préfixe*, *facteur* et *suffixe* de u .

Exemple 3 *Les mots abab, baba et bababab... sont respectivement un préfixe, un facteur et un suffixe du mot abababab...*

Un ensemble quelconque de mots est un *langage*. L'ensemble vide, noté \emptyset est le langage ne contenant aucun mot.

Si Γ et Σ sont deux alphabets, on appelle *morphisme* une application δ de Γ vers Σ^* vérifiant :

$$\delta(u \cdot v) = \delta(u) \cdot \delta(v)$$

ce qui implique $\delta(\varepsilon) = \varepsilon$.

À partir de ces quelques définitions nous allons pouvoir construire nos premières familles de mots infinis.

Si v est un mot fini, le mot infini $w = v^\omega$ est un *mot périodique*. Si u et v sont deux mots finis, alors le mot infini $w = uv^\omega$ est un *mot ultimement périodique*.

Soit Γ un alphabet, un *mot morphique* est un mot de la forme :

$$\sigma(\tau^\omega(a)) = \sigma(a\tau(u) \dots \tau(u)^n \dots)$$

où σ et τ sont des morphismes de Γ dans Γ^* avec $a \in \Gamma$ et $\tau(a) = au$.

Exemple 4 *Le mot infini ababab... est le mot ab^ω .*

Le mot infini aabab²ab³... est le mot morphique obtenu à l'aide des morphismes suivants :

$$\begin{aligned} \sigma : \Delta &\mapsto \varepsilon, a \mapsto a, b \mapsto b \\ \tau : \Delta &\mapsto \Delta a, a \mapsto ab, b \mapsto b \end{aligned}$$

1.2 Graphes

Nous allons maintenant définir les graphes, qui seront les outils de base de tous les travaux présentés après.

Définition 1.1 Un graphe étiqueté est un triplet (V, Σ, G) composé des trois ensembles suivants :

- un ensemble V de sommets,
- un alphabet fini Σ d'étiquettes,
- un ensemble $G \subseteq V \times \Sigma \times V$ d'arcs.

Quand V et Σ sont implicites, on assimilera le graphe à G . Un triplet (s, a, t) de G est un arc de *source* s , de *destination* t et d'*étiquette* a , que l'on notera $s \xrightarrow[G]{a} t$ ou simplement $s \xrightarrow{a} t$ si G est implicite.

On notera V_G (resp. Σ_G) l'ensemble des sommets (resp. étiquettes) d'un graphe G .

Définition 1.2 Soit Γ un nouvel alphabet de couleurs. Un graphe coloré, est un graphe tel que $G \subseteq V \times T \times V \cup \Gamma \times V$.

Un doublet (c, s) de G est un sommet s , coloré par c .

Un graphe est *fini* si il a un nombre fini de sommets. Un graphe est *déterministe* si deux arcs distincts de même source ont des étiquettes différentes.

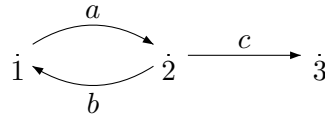


FIG. 1.1 – Un graphe étiqueté et coloré.

Exemple 5 Le graphe de la figure 1.1 est un graphe étiqueté sur l'alphabet $\{a, b, c\}$ et coloré sur l'alphabet $\{1, 2, 3\}$. Il est à la fois fini et déterministe.

Un *chemin* de s à t est une suite finie d'arcs $(s, u_1, s_1) \dots (s_{n-1}, u_n, t)$ dont la suite des étiquettes forment le mot $u = u_1 \dots u_n$, que l'on notera $s \xrightarrow[G]{u} t$ ou plus simplement $s \xrightarrow{u} t$ si G est implicite.

Définition 1.3 Deux graphes G et G' sont isomorphes si il existe une bijection f de V_G sur $V_{G'}$ telle que :

$$\forall s, t \in V_G, \forall a \in T, \quad s \xrightarrow[G]{a} t \iff f(s) \xrightarrow[G']{a} f(t).$$

On appellera *degré entrant* (resp. *sortant*) d'un sommet s le nombre d'arcs aboutissants à (*issus de*) ce sommet. Le *degré* d'un sommet étant la somme de ses degrés entrant et sortant.

Deux sommets s et t d'un graphe G sont dit *voisins* si il existe une étiquette $a \in \Sigma$ telle que l'on ait soit l'arc (s, a, t) soit l'arc (t, a, s) dans

G . Deux sommets sont dit *connectés* si il existe dans G une suite finie de sommets s_1, \dots, s_n telle que les paires de sommets (s, s_1) , (s_n, t) et $\forall 1 \leq i \leq n-1, (s_i, s_{i+1})$ soient connectés. Un graphe est dit *connexe* si toute paire de sommets est connectée.

L'inverse d'un graphe G , noté G^{-1} est le graphe :

$$G^{-1} = \{(t, a, s) \mid (s, a, t) \in G\}.$$

La *distance* entre deux sommets s et s de G notée $d_G(s, t)$, est la longueur du plus court chemin allant de s à t dans le graphe $G \cup G^{-1}$.

La *restriction* d'un graphe G , par rapport à un sommet r et une distance n , est le graphe :

$$G_{r,n} = \{s \xrightarrow{a} t \mid s \xrightarrow{a} t \in G \wedge d_G(r, s) \leq n \wedge d_G(r, t) \leq n\}.$$

L'objectif étant d'utiliser les mots pour caractériser des familles de graphes, nous allons maintenant définir les relations entre certains graphes et les mots.

Définition 1.4 *Un graphe connexe fini dont tout sommet est de degré entrant 1 excepté un sommet s de degré entrant 0, et dont tout sommet est de degré sortant 1 excepté un sommet t de degré sortant 0, représente le mot fini u étiquetant le chemin $s \xrightarrow{u} t$. En particulier $u = \varepsilon$ pour $s = t$.*

On peut étendre la définition précédente aux graphes infinis.

Définition 1.5 *Un graphe infini connexe dont tout sommet est de degré entrant 1 excepté un sommet s de degré entrant 0, et dont tout sommet est de degré sortant 1, représente le mot infini u étiquetant le chemin allant de s et se poursuivant à l'infini.*

Remarque 1 *Par abus de langage, on utilisera le terme mot pour désigner aussi bien la suite de lettres que le graphe qui la représente (à isomorphisme près) lorsqu'il n'est pas nécessaire de différencier les deux.*

1.3 Arbres

Nous allons maintenant définir une catégorie particulière de graphes à laquelle nous nous intéresserons particulièrement dans la suite de ce rapport, à savoir les arbres.

Définition 1.6 Un arbre est un graphe connexe ayant un sommet de degré entrant nul : sa racine, et dont tous les autres sommets ont un degré entrant égal à 1.

Un arbre est *fini* si son graphe est fini. Un arbre est *déterministe* si son graphe est déterministe.

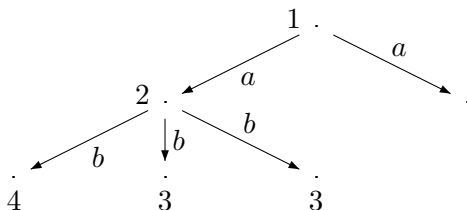


FIG. 1.2 – Un arbre fini.

Exemple 6 Le graphe de la figure 1.2 est un arbre fini déterministe qui a pour racine le sommet coloré par 1.

Tout sommet d'un graphe est appelé un *nœud*. Tout nœud de degré sortant 0 est une *feuille*. On appellera *père* d'un nœud s l'unique nœud qui est source de l'arc qui a pour destination s (à l'exception de la racine qui n'a pas de sommet père) et *fil*s d'un nœud s , tout nœud ayant s pour père.

L'ensemble des *descendants* d'un nœud s , est l'ensemble des nœuds t tels qu'il existe un chemin de s à t . L'ensemble des *ancêtres* d'un nœud s est l'ensemble des nœuds ayant s comme descendant.

Le *sous-arbre* d'un nœud s est l'arbre restreint aux descendants de s .

Un *prefixe* A' d'un arbre A , est un arbre que l'on peut obtenir à partir de A en supprimant un sous-ensemble de nœuds ainsi que tous leurs descendants.

Définition 1.7 Un arbre est ordonné si pour tout nœud s , on dispose d'une relation d'ordre sur les fils de s .

Sur un arbre ordonné, on dira qu'un nœud s est à *gauche* (*resp.* à *droite*) d'un nœud t , si parmi leurs ancêtres on peut trouver deux nœuds, s' ancêtre de s et t' ancêtre de t , ayant le même père et tel que s' soit inférieur (*resp.* supérieur) à t' .

On peut étendre cette définition aux sous-arbres. Pour tout arbre ordonné, on dira qu'un sous-arbre A' est à gauche (*resp.* à droite) d'un sous-arbre A'' , si la racine de A' est à gauche (*resp.* à droite) de la racine de A'' .

On appellera *branche gauche* (*resp.* *droite*) d'un arbre ordonné le plus petit ensemble de nœuds contenant la racine ainsi que le plus petit (*resp.* grand) fils de chaque sommet de l'ensemble.

La *branche d'une feuille* est le plus petit ensemble contenant cette feuille ainsi que tous les sommets dont l'unique fils est dans cet ensemble.

Exemple 7 Sur l'arbre de la figure 1.2, le nœud de couleur 2 a pour père le nœud de couleur 1 et pour fils les nœuds coloriés par 3 et 4. Le nœud de couleur 4 est à gauche des nœuds de couleur 3. La branche gauche est constituée des nœuds coloriés par 1, 2 et 3.

Ces définitions vont maintenant nous permettre de définir un mot d'un arbre comme étant le mot de ses feuilles.

Définition 1.8 On appelle mot des feuilles d'un arbre ordonné A , soit le mot vide si la branche gauche de A est infinie, soit le mot formé par la concaténation de la couleur de la feuille la plus à gauche de A et du mot des feuilles de l'arbre A privé de la branche de cette feuille.

Propriété 1 Soit A un arbre ordonné. Si il existe un sous-arbre infini A' de A tel qu'il n'existe pas d'autre sous-arbre infini à gauche de A' dans A , alors le mot des feuilles de A' est un suffixe du mot des feuilles de A .

1.4 Automates

1.4.1 Automates généraux

Définition 1.9 Un automate est un triplet (G, I, F) composé de :

- un graphe G ,
- un ensemble de sommets initiaux I ,
- un ensemble de sommets terminaux F .

Remarque 2 Pour les automates, on appelle les sommets du graphe des états ; on parlera donc d'états initiaux et d'états terminaux.

Pour tout automate \mathcal{A} de graphe G , si G est fini alors \mathcal{A} est un *automate fini*, si G est déterministe alors \mathcal{A} est un *automate déterministe*.

Un mot u est *reconnu* par un automate \mathcal{A} , si u étiquette un chemin sur G partant d'un état initial et arrivant à un état final.

On appelle *langage reconnu* par un automate \mathcal{A} , l'ensemble des mots reconnus par cet automate.

Exemple 8 Pour le graphe de la figure 1.1, si l'on prend comme état initial le sommet colorié par 1 et source de l'arc étiqueté par a , et comme état terminal l'autre sommet colorié par 1, on définit un automate reconnaissant le langage $a(ab)^*c$.

Les automates permettent de définir des langages ou des familles de langages. On doit notamment le résultat suivant à *Kleene* :

Définition 1.10 ([Kle56]) *Les automates finis reconnaissent exactement les langages réguliers.*

Ce résultat met en évidence qu'il n'est pas possible de définir tous les langages avec les automates finis. Ainsi le langage $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas régulier. Nous allons voir qu'il est par contre possible de le définir à l'aide d'un automates infini.

On appelle *résidus à gauche* d'un langage L par un mot fini $u \in \Sigma$, le langage : $u^{-1}L = \{v \mid uv \in L\}$.

On peut maintenant, à l'aide des résidus à gauche d'un langage L définir un automate \mathcal{A} , fini ou infini, reconnaissant ce langage. On commence par définir le graphe des résidus :

$$\vec{L} = \{u^{-1}L \xrightarrow{a} (ua)^{-1}L \mid u \in \Sigma^* \wedge a \in \Sigma\}.$$

L'automate des résidus \mathcal{A} , qui reconnaît le langage L , est défini par :

$$\mathcal{A}(L) = (\vec{L}, \{L\}, \{L' \mid \varepsilon \in L'\}).$$

Exemple 9 *Le langage $\{a^n b^n \mid n \geq 0\}$ a pour graphe des résidus le graphe de la figure 1.3.*

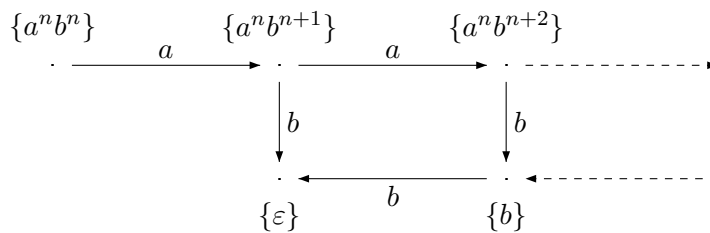


FIG. 1.3 – Graphe des résidus de $\{a^n b^n \mid n \geq 0\}$.

1.4.2 Automates à pile

Nous venons de définir, à la fois des automates généraux et une méthode pour construire un automate déterministe reconnaissant n'importe quel langage. Mais les automates infinis sont des objets difficiles à manipuler et peut utile en pratique. Nous allons donc rappeler une nouvelle famille d'automates qui est plus puissante que les automates finis.

Définition 1.11 ([Har78, HU79]) Un automate à pile est un quadruplet (Q, Λ, Σ, R) composé des ensembles suivants :

- un ensemble finis Q d'états,
- un alphabet de pile Λ ,
- un alphabet de bande Σ ,
- un ensemble de règles $R \subseteq Q \times \Lambda \times \Sigma \times Q \times \Lambda^*$.

Une règle (p, M, a, q, N) est notée $pM \xrightarrow{a} qN$. Une configuration d'un automate à pile est un couple (o, U) avec $o \in Q$ et $U \in \Lambda^*$ où o est l'état courant et U le mot de pile de l'automate.

Soit un automate \mathcal{A} de configuration (o, U) , avec $U = U_0 \dots U_n$, et une règle $r = pM \xrightarrow{a} qN$. La règle r est applicable si $o = p$, si $U_n = M$ et si la lettre lue sur la bande est a . L'application de la règle r fait passer l'automate dans la configuration $(q, U_0 \dots U_{n-1}N)$.

Un automate à pile reconnaît le mot qu'il lit sur la bande. Un automate peut reconnaître sur pile vide, auquel cas lorsque la pile est vide le mot lu est reconnu ; il peut aussi reconnaître sur état final, auquel cas le mot lu est reconnu si l'état courant est un état final, ou bien à la fois sur pile vide et sur état final.

Ces automates nous apportent une puissance supplémentaire par rapport aux automates finis ; ils sont par exemple capables de reconnaître le langage $\{a^n b^n \mid n \in \mathbb{N}\}$ qui n'est pas régulier. Rappelons un résultat classique sur les langage reconnus par les automates à piles.

Définition 1.12 Les automates à pile reconnaissent exactement les langages algébriques.

Exemple 10 Le langage $L = a^+ \{b^n c^n \mid n \geq 1\}$ est reconnu sur pile vide par l'automate à pile de la figure 1.4.

$$\mathcal{A} \left| \begin{array}{lll} pA \xrightarrow{a} pB & pB \xrightarrow{b} qBC & qC \xrightarrow{c} q \\ pB \xrightarrow{a} pB & pB \xrightarrow{b} qC & \\ & qB \xrightarrow{b} qBC & \\ & qB \xrightarrow{b} qC & \end{array} \right.$$

FIG. 1.4 – Exemple d'automate à pile.

1.5 Transducteurs

Nous allons maintenant définir une dernière famille de structures proche des automates, à savoir les transducteurs. Ils n'engendrent pas directement des mots, mais des graphes. Ces graphes pouvant être des mots, ils est intéressant de les étudier.

Définition 1.13 ([Pri00]) Un transducteur T est sextuplet $(Q, \Gamma, \Sigma, G, I, F)$ composé des ensembles :

- un ensemble fini d'états Q ,
- un alphabet d'étiquettes Γ ,
- un alphabet de sortie Σ ,
- un ensemble fini d'arcs $G \subseteq Q \times \Gamma \cup \{\varepsilon\} \times \Gamma \cup \{\varepsilon\} \times Q$,
- un ensemble d'états initiaux $I \subseteq Q$,
- un ensemble d'états finaux $F \subseteq Q$,
- une fonction terminale $\delta : F \rightarrow \Sigma$.

De même que pour les graphes, une transition (p, a, b, q) sera notée $p \xrightarrow{a/b} q$.

On définit aussi, de la même manière que pour les graphes, un chemin $p \xrightarrow{u/v} q$ avec $u, v \in \Gamma^*$ comme suit :

$$p \xrightarrow{u_1/v_1} p_1 \dots p_n \xrightarrow{u_n/v_n} q \text{ avec } a = \delta(p_n), u = u_1 \dots u_n \text{ et } v = v_1 \dots v_n.$$

Un transducteur T est dit *synchronisé à gauche* (resp. à droite) si pour tout chemin $p \xrightarrow{u/v} q$, il existe $0 \leq m \leq n$ tel que $\forall i \leq m, u_i \neq \varepsilon \wedge \forall i > m, u_i = \varepsilon$. (resp. $\forall i \leq m, v_i \neq \varepsilon \wedge \forall i > m, v_i = \varepsilon$)

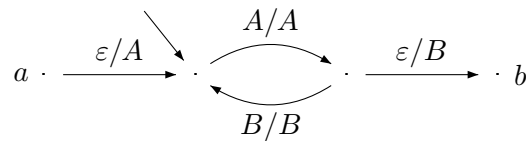


FIG. 1.5 – Un transducteur synchronisé à gauche.

Les transducteurs vont nous servir à définir des familles de graphes. Nous allons donc maintenant présenter comment un transducteur caractérise un graphe.

On dit qu'un arc $s \xrightarrow{a} t$ d'un graphe G est reconnu par un transducteur T s'il existe un chemin sur $p \xrightarrow{s/t} q$ allant d'un initial p à un état final q tel

que $\delta(q) = a$.

Ceci nous amène à la définition suivante d'un graphe engendré par un transducteur.

Définition 1.14 *Un graphe G est engendré par un transducteur T si T reconnaît exactement les arcs de G .*

Exemple 11 *Le graphe de la figure 1.6 est reconnu par le transducteur de la figure 1.5.*

$$\varepsilon \xrightarrow{a} A \xrightarrow{a} AB \xrightarrow{a} ABA \dashrightarrow$$

FIG. 1.6 – Un graphe synchronisé.

Chapitre 2

La hiérarchie

Nous allons maintenant pouvoir définir des familles de graphes infinis et leurs propriétés avant de voir pourquoi ces graphes sont intéressants. Nous présenterons enfin la hiérarchie de graphes, et les familles qui ont été étudiées durant ce stage.

2.1 Familles de graphes infinis

2.1.1 Une première famille de graphes infinis

On peut maintenant présenter une première famille de graphes infinis, qui va représenter les exécutions possibles des automates à piles.

A partir des automates à pile, Muller et Schupp ont défini en 1985 [MS85] la famille des graphes des automates à pile, qui à chaque automate à pile associe le graphe suivant :

$$G(\mathcal{A}) = \{ pUW \xrightarrow{a} qVW \mid (pU \xrightarrow{a} qV) \in \mathcal{A} \wedge W \in P^* \}.$$

De plus, tout graphe d'automate à pile est de degré borné : il existe une borne sur les degrés des sommets. Remarquons une autre propriété de base.

Propriété 2 *Tout graphe d'automate à pile n'a qu'un nombre fini de composantes connexes (à isomorphisme près).*

Pour pouvoir inclure aussi les graphes finis, on introduit la notion de *restriction rationnelle* sur les graphes. Soit $G(\mathcal{A})$ le graphe d'un automate à pile et $L \in \text{Rat}(QP^*)$ un langage rationnel. On appelle $G(\mathcal{A})|_L$ le graphe restreint au langage L :

$$G(\mathcal{A})|_L = \{ pUW \xrightarrow{a} qVW \mid (pU \xrightarrow{a} qV) \in \mathcal{A} \wedge W \in P^* \wedge pUW, qVW \in L \}.$$

Par restriction rationnelle, on peut isoler une composante connexe.

Propriété 3 *Les restrictions rationnelles des graphes des automates à pile qui sont connexes sont exactement les composantes connexes des graphes des automates à pile.*

On peut ainsi donner le graphe de l'automate de la figure 1.4 :

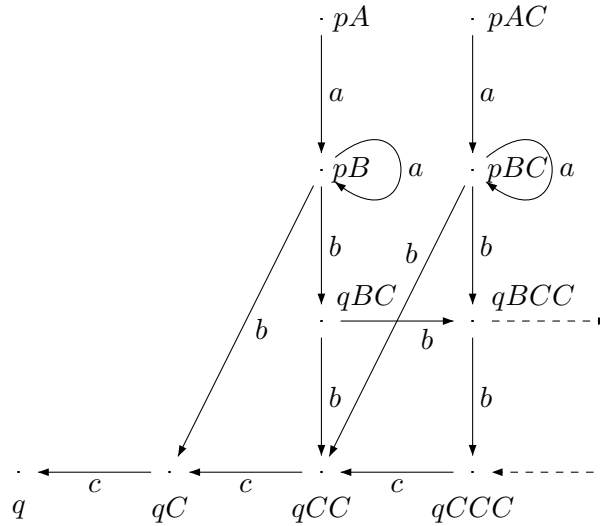


FIG. 2.1 – Graphe de l'automate à pile de la figure 1.4.

2.1.2 Graphes réguliers

Nous allons maintenant définir une autre famille de graphes, en utilisant une approche géométrique. Puis nous montrerons les liens entre cette famille et la précédente.

La récriture de graphes nous donne une autre manière de définir des graphes infinis. Elle consiste à généraliser la récriture de mots aux graphes, et permet à l'aide de grammaires de graphes d'engendrer des graphes infinis, appelés graphes *réguliers* ou *HR-équationnels* [Cou90].

On cherche à définir des graphes à l'aide de systèmes de récriture. Soit N un nouvel alphabet. On appelle *hyperarc terminal* (resp. *non-terminal*) un arc étiqueté par une lettre de Σ (resp. N). Une règle de récriture associée à un hyperarc non-terminal un hypergraphe (composé d'hyperarcs terminaux et non-terminaux).

En partant d'un hypergraphe axiome, un pas de récriture consiste à remplacer dans le graphe, un hyperarc non-terminal par l'hypergraphe correspondant.

La grammaire de la figure 2.2 engendre à partir de l'hyperarc A le graphe de la figure 2.3.

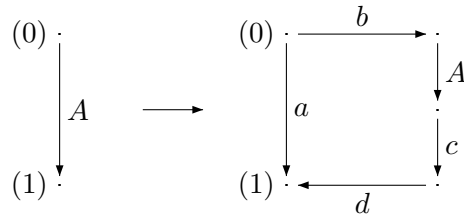


FIG. 2.2 – Exemple de grammaire de graphes.

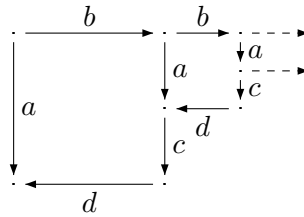


FIG. 2.3 – Graphe engendré par la grammaire précédente.

À partir d'un graphe fini, appelé axiome, une grammaire de graphes engendre par récritures parallèles itérées un seul graphe à isomorphisme près, appelé *graphe régulier*.

Une composante connexe H de $G_{v,m}$ est *frontière isomorphe* à une composante connexe K de $G_{v,n}$ s'il existe un isomorphisme f de H sur K tel que $d(v,t) = m \iff d(v,f(t)) = n$ pour tout $t \in V_H$.

Un graphe connexe est *finiment décomposable par distance* à partir d'un sommet s si il n'a qu'un nombre fini de composantes connexes à frontière isomorphisme près sur les $G_{s,n}$ pour tout $n \geq 0$.

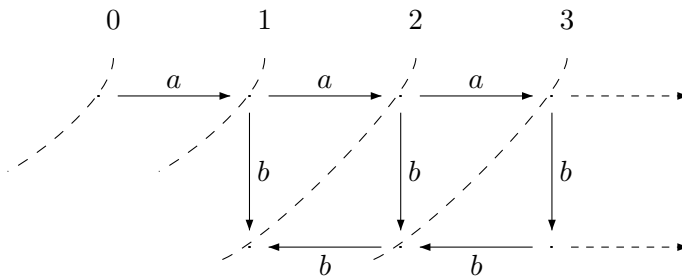


FIG. 2.4 – Graphe finiment décomposable par distance.

Si un graphe est finiment décomposable à partir d'un sommet, alors il est finiment décomposable à partir de tout sommet.

Propriété 4 ([Cau95]) *Un graphe est régulier si et seulement si il n'a qu'un nombre fini de composantes connexes à isomorphisme près, et que chaque composante est finiment décomposable par distance.*

On peut maintenant faire le lien avec les graphes des automates à pile à l'aide du résultat ci-dessous.

Propriété 5 *Toute composante connexe d'un graphe d'automate à pile est un graphe régulier.*

Nous sommes en mesure d'énoncer un résultat essentiel, dû à Muller et Schupp.

Théorème 2.1 ([MS85]) *Les restrictions rationnelles des graphes des automates à pile sont, à isomorphisme près, les graphes réguliers de degré borné.*

2.1.3 Graphes préfixe reconnaissables

Un *système de réécriture* de mots avec étiquetage, est un système de règles de la forme : $U \xrightarrow[R]{a} V$ avec $U, V \in N^*$ et $a \in T$. Le *graphe des réécritures* de R est $\{gUd \xrightarrow{a} gVd \mid U \xrightarrow[R]{a} V \wedge g, d \in N^*\}$.

Le *graphe de réécriture préfixe* de R (où *graphe préfixe-reconnaissable*) est le graphe $\{Ud \xrightarrow{a} Vd \mid U \xrightarrow[R]{a} V \wedge d \in N^*\}$. Ils sont liés aux autres familles de graphes par les propriétés ci-dessous.

Propriété 6 *On a équivalence entre*

les composantes connexes des graphes de réécriture préfixe,
les composantes connexes des graphes des automates à pile,
les graphes réguliers connexes de degré finis.

Propriété 7 ([Cau95]) *On a équivalence entre*

les restrictions rationnelles des graphes de réécriture préfixe,
les restrictions rationnelles des graphes des automates à pile,
les graphes réguliers de degré finis.

2.2 Théories logiques

Nous venons de définir des familles de graphes infinis dont une famille représentant les exécutions des graphes des automates à pile. Il est intéressant de pouvoir vérifier des propriétés dessus. Nous allons ici définir une logique qui nous permettra d'exprimer ces propriétés.

On cherche à vérifier des propriétés sur les graphes que l'on vient de définir. Pour cela, on va définir des formules logiques permettant d'exprimer ces propriétés. Il faudra ensuite décider de la véracité d'une formule φ sur un graphe G . Dans un premier temps, nous allons définir la logique au premier ordre. Mais celle-ci étant de faible puissance, nous l'étendrons ensuite à la logique au deuxième ordre monadique.

2.2.1 Logique au premier ordre

Soient \mathcal{V} un ensemble dénombrable de variables de sommets : x, y, z, \dots et Σ un alphabet. La logique au premier ordre a deux *formules atomiques* : l'égalité $x = y$ et l'existence d'un arc $x \xrightarrow{a} y$, où $a \in \Sigma$ est une étiquette. On construit ensuite l'ensemble des formules \mathcal{F} à l'aide des connecteurs logiques \vee, \neg et \exists selon les règles suivantes :

- toute formule atomique est une formule,
- si φ et ψ sont des formules alors $\varphi \vee \psi$ et $\neg\varphi$ sont aussi des formules,
- si φ est une formule et x est une variable alors $\exists x.\varphi$ est aussi une formule.

L'ensemble des *variables libres* d'une formule φ est défini par les règles suivantes :

$$\begin{aligned} Libre(x) &= \{x\} \\ Libre(x = y) &= \{x, y\} \\ Libre(x \xrightarrow{a} y) &= \{x, y\} \\ Libre(\varphi \vee \psi) &= Libre(\varphi) \cup Libre(\psi) \\ Libre(\neg\varphi) &= Libre(\varphi) \\ Libre(\exists x.\varphi) &= Libre(\varphi) - \{x\} \end{aligned}$$

Soit $\varphi(x_1, \dots, x_n)$ la formule φ ayant x_1, \dots, x_n comme variables libres. On va chercher à interpréter φ sur un graphe G à l'aide du système de règles ci-dessous : pour x, y, x_1, \dots, x_n des variables de la formule et s, t, s_1, \dots, s_n des sommets de G ,

$$\begin{aligned} [x \leftarrow s, y \leftarrow t](x = y) &= \begin{cases} 1 \text{ si } s = t \\ 0 \text{ sinon} \end{cases} \\ [x \leftarrow s, y \leftarrow t](x \xrightarrow{a} y) &= \begin{cases} 1 \text{ si } s \xrightarrow{a}_G t \\ 0 \text{ sinon} \end{cases} \\ [\sigma](\varphi \vee \psi) &= [\sigma_i](\varphi) \vee [\sigma_j](\psi) \\ [\sigma](\neg\varphi) &= \neg([\sigma](\varphi)) \\ [\sigma](\exists x.\varphi) &= \begin{cases} 1 \text{ si } \exists s \in V(G) [\sigma, x \leftarrow s](\varphi) = 1 \\ 0 \text{ sinon} \end{cases} \end{aligned}$$

avec :

$$\begin{aligned} \sigma &= x_1 \leftarrow s_1, \dots, x_n \leftarrow s_n \\ \sigma_i &= x_{i_1} \leftarrow s_{i_1}, \dots, x_{i_p} \leftarrow s_{i_p} \quad i_1 < \dots < i_p \text{ et } \{x_{i_1}, \dots, x_{i_p}\} = Libre(\varphi) \\ \sigma_j &= x_{j_1} \leftarrow s_{j_1}, \dots, x_{j_q} \leftarrow s_{j_q} \quad j_1 < \dots < j_q \text{ et } \{x_{i_1}, \dots, x_{i_p}\} = Libre(\psi). \end{aligned}$$

On note $G \models \varphi(s_1, \dots, s_n)$ le fait que G vérifie la formule φ :

$$[x_1 \leftarrow s_1, \dots, x_n \leftarrow s_n]\varphi = 1.$$

On définit l'ensemble des *instances positives* d'une formule sur un graphe par :

$$\llbracket \varphi(x_1, \dots, x_n) \rrbracket_G = \{(s_1, \dots, s_n) \mid s_1, \dots, s_n \in V_G \wedge G \models \varphi(s_1, \dots, s_n)\}.$$

Si on se restreint aux *formules closes* (ou *énoncés*), c'est-à-dire les formules φ telles que $\text{Libre}(\varphi) = \emptyset$, alors on a :

$$\llbracket \varphi \rrbracket_G = \begin{cases} \{()\} & \text{si } G \models \varphi \\ \emptyset & \text{sinon.} \end{cases}$$

La logique au premier ordre permet de vérifier des propriétés locales sur les graphes. Par exemple, il est possible de vérifier que tout sommet est source d'un arc étiqueté par a avec la formule $\forall x, \exists y, x \xrightarrow{a} y$.

2.2.2 Logique au second ordre monadique

La logique au premier ordre est limitée aux propriétés locales. Pour pouvoir exprimer des propriétés telles que l'accessibilité, il est nécessaire de l'étendre. La logique au second ordre monadique introduit des variables d'ensembles de sommets et permet de vérifier des propriétés globales sur le graphe.

On la définit en ajoutant à la logique du premier ordre des variables d'ensembles : X, Y, \dots , un axiome d'appartenance : $x \in X$ qui vérifie que la variable de sommet x appartient à l'ensemble X de sommets et la règle de construction de formules :

si X est une variable d'ensemble et φ une formule alors $\exists X.\varphi$ est une formule.

L'ensemble des variables libres d'une formule est définie de la même manière que pour une formule du premier ordre.

On peut alors exprimer l'accessibilité comme suit :

$$\begin{aligned} \text{Ferme}(X) &: \forall x \forall y (x \in X \wedge x \longrightarrow y \implies y \in X) \\ x \longrightarrow^* y &: \forall X ((x \in X \wedge \text{Ferme}(X)) \implies y \in X). \end{aligned}$$

On appelle *théorie monadique* d'un graphe G , l'ensemble des formules closes satisfaites par G :

$$\text{ThM}(G) = \{\varphi \mid \text{Libre}(\varphi) = \emptyset \wedge G \models \varphi\}.$$

Le premier résultat de décidabilité de la logique au second ordre monadique sur les graphes est dû à Büchi.

Définition 2.1 [Bü60] *La demi-droite a une théorie monadique décidable.*

Ce résultat est généralisé par Rabin à l'arbre binaire complet :

Définition 2.2 [Rab69] *L'arbre binaire complet a une théorie monadique décidable.*

Ce résultat a été étendu par Muller et Schupp aux graphes des automates à pile en 1985.

Définition 2.3 [MS85] *Les graphes des automates à pile ont une théorie monadique décidable.*

À partir de ces premiers graphes ayant une théorie monadique décidable, on va chercher à obtenir des familles générales de graphes à l'aide de transformations préservant la décidabilité.

Une *interprétation monadique* est une transformation logique qui à partir d'un graphe G fait correspondre un graphe $I(G) = \{s \xrightarrow{a} t \mid G \models I_a(s, t)\}$ avec $\forall a \in T, I_a(x, y)$ est une formule monadique ayant deux variables libres de sommets.

Pour un graphe coloré, on étend la définition : $I(G) = \{s \xrightarrow{a} t \mid G \models I_a(s, t)\} \cup \{cs \mid G \models I_c(s)\}$ avec $\forall c \in \Gamma, I_c(x)$ est une formule monadique ayant une variable libre de sommet.

Propriété 8 *Soit G un graphe ayant une théorie monadique décidable et I une interprétation monadique. Le graphe $I(G)$ a une théorie monadique décidable.*

2.3 Transformations de graphes

Nous venons donc de présenter une famille de graphes sur laquelle il est possible de décider des formules logiques monadiques. Ces formules nous permettent aussi de définir de nouveaux graphes infinis sur lesquels nous pouvons toujours décider de ces formules. Mais une autre approche, plus structurelle, va nous permettre de définir une infinité de familles de graphes infinis, tout en conservant cette décidabilité.

Une *application rationnelle inverse* est une transformation qui à un graphe G fait correspondre le graphe $S(G) = \{s \xrightarrow{a} t \mid \exists u \in S_a, s \xrightarrow{u} t\}$ avec $\forall a \in \Sigma, S_a \in \text{Rat}(\Sigma^*)$.

Afin de pouvoir suivre les arcs en sens inverse, on ajoute les étiquettes inverse sur les arcs. On se donne un nouvel alphabet $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$ en bijection avec Σ . Une transition de la forme $s \xrightarrow{\bar{a}} t$ signifie que $t \xrightarrow{a} s$ est un arc de G . On étend alors la définition d'un chemin $s \xrightarrow{u} t$ aux mots $u \in (\Sigma \cup \bar{\Sigma})^*$.

On redéfinit alors les applications rationnelles inverses en utilisant les étiquettes inverses ainsi que les couleurs : $\forall a \in \Sigma, S_a \in \text{Rat}((\Sigma \cup \bar{\Sigma} \cup \Gamma)^*)$.

Cette transformation est une interprétation monadique, et préserve donc la décidabilité de la théorie monadique du second ordre.

Une *application finie inverse* est une transformation similaire à la précédente, à la différence que l'on utilise les langages finis à la place des langages rationnels : $\forall a \in \Sigma, S_a \in Fin(\Sigma^*)$.

Propriété 9 ([Cau96]) *L'application rationnelle inverse et l'application finie inverse sont des interprétations monadiques, et préservent donc la décidabilité de la théorie monadique du second ordre.*

Le *dépliage* d'un graphe G à partir d'un sommet r , est l'arbre de tous les chemins partant de r sur G . Ainsi $Ux \xrightarrow{a} Uxay$ est un arc du dépliage de G si $x \xrightarrow{a} y$ est un arc de G et $U \in (V_G \Sigma_G)^*$.

$$Dep(G, r) = \{ra_1r_1 \dots a_n r_n \xrightarrow{a} ra_1r_1 \dots a_n r_n a_{n+1} r_{n+1} \mid n \geq 0 \wedge r \xrightarrow{a_1} r_1 \dots \xrightarrow{a_n} r_n \xrightarrow{a_{n+1}} r_{n+1}\}.$$

Si le graphe G est déterministe on peut le simplifier en :

$$Dep(G, r) = \{u \xrightarrow{a} ua \mid \exists t, r \xrightarrow{ua} t\}$$

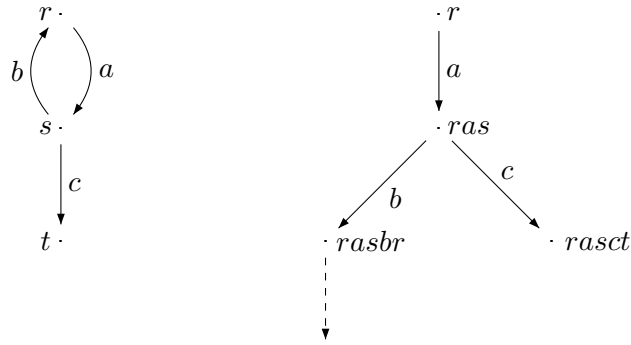


FIG. 2.5 – Graphe et son arbre de dépliage à partir du sommet r .

Contrairement à l'application rationnelle inverse, le dépliage n'est pas une interprétation monadique. Mais la décidabilité de la théorie monadique reste préservée.

Définition 2.4 ([CW98]) *Le dépliage est une transformation de graphes qui préserve la décidabilité de la logique du second ordre monadique.*

L' ε -cloture d'un graphe coloré pour une étiquette $e \in T$ et une couleur $\# \in \Gamma$ est :

$$(e, \#)\text{-cloture}(G) = \{s \xrightarrow{a} t \mid a \in \Sigma - \{e\} \wedge s \xrightarrow[G]{e^*ae^*} t \wedge \#s, \#t \in G\} \cup \{fs \mid fs \in G \wedge f \neq \#\}.$$

Définition 2.5 L' ε -cloture est une application rationnelle inverse et préserve donc la décidabilité de la théorie monadique.

2.4 La hiérarchie

Les transformations précédentes vont maintenant nous permettre de définir la hiérarchie de graphes infinis qui est étudiée dans ce stage.

L'application de la transformation Rat^{-1} à la famille des arbres finis, nous donne les graphes finis. Si ensuite on applique le dépliage au graphes finis, on obtient une nouvelle famille d'arbres : les arbres réguliers de degré fini. Ces deux transformations appliquées l'une après l'autre de manière répétée, nous permettent de définir la hiérarchie de familles de graphes [Cau02] de la figure 2.6.

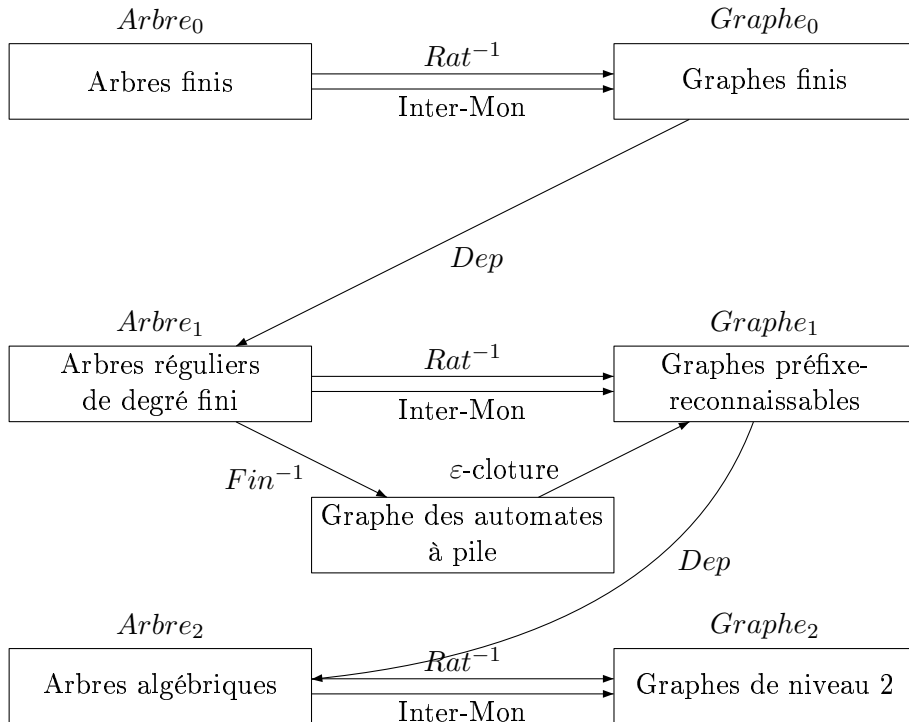


FIG. 2.6 – Une hiérarchie de graphes.

Les familles de graphes obtenues par Rat^{-1} ou par interprétation monadique sont closes par ces deux transformations. Il est possible d'engendrer

les graphes des automates à pile à partir des arbres réguliers de degré fini, en utilisant la transformation Fin^{-1} . On obtient ensuite les graphes préfixe-reconnaissables à l'aide de l' ε -cloture.

Une transformation $* : G \mapsto G^*$ est *monadique compatible* si on peut transformer toute formule monadique close φ en une autre formule φ^* , de sorte que $G^* \models \varphi \iff G \models \varphi^*$. Une transformation monadique compatible préserve la décidabilité de la théorie monadique : $ThM(G)$ décidable $\implies ThM(G^*)$ décidable.

Les transformations vues précédemment (interprétation monadique, dépliage, ...) sont toutes monadique compatible. Il est donc possible de décider une formule logique au deuxième ordre monadique sur n'importe quel graphe de la hiérarchie.

Chapitre 3

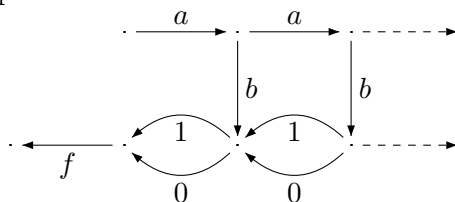
Mots infinis dans la hiérarchie

Avant de nous intéresser à des familles générales de mots infinis, nous allons commencer par situer dans la hiérarchie quelques mots infinis particuliers.

3.1 Mot de Champernowne

Le mot de Champernowne est la concaténation de tous les entiers, pris dans l'ordre, de 0 à l'infini : 012...91011...1920.... Dans la suite nous utiliserons la notation en base 2, mais la preuve se généralise dans toutes les bases.

On part du graphe G suivant :



On a $G \in \text{Graphe}_1$. On prend r la racine de G , et on construit le graphe $H = g^{-1}(\text{Dep}(G, r))$ où g est l'application rationnelle suivante :

$$\begin{aligned} a &\mapsto a & ; & & 1 &\mapsto 1 \\ b &\mapsto b & ; & & 0 &\mapsto \bar{n}n0 & \quad (\forall n \in \{0, 1\}) \\ f &\mapsto f & ; & & f &\mapsto f \\ b' &\mapsto \bar{b} & ; & & n' &\mapsto \bar{n} & \quad (\forall n \in \{0, 1\}) \end{aligned}$$

Donc $H \in \text{Graphe}_2$ et nous construisons le graphe $K = h^{-1}(\text{Dep}(H, r))$ où h est l'application rationnelle suivante :

$$\begin{aligned} a &\mapsto ab0f\bar{f}0' \\ b &\mapsto f\bar{f}1^*(\bar{0}1)0^*f\bar{f}(0'+1')^*b'\bar{b}' \\ 0 &\mapsto 0' \\ 1 &\mapsto 1' \end{aligned}$$

suivie de :

$$\begin{aligned} 0 &\mapsto 0\bar{0}0(\bar{b} + \bar{0}0 + \bar{1}1) \\ 1 &\mapsto 1\bar{1}1(\bar{b} + \bar{0}0 + \bar{1}1) \end{aligned}$$

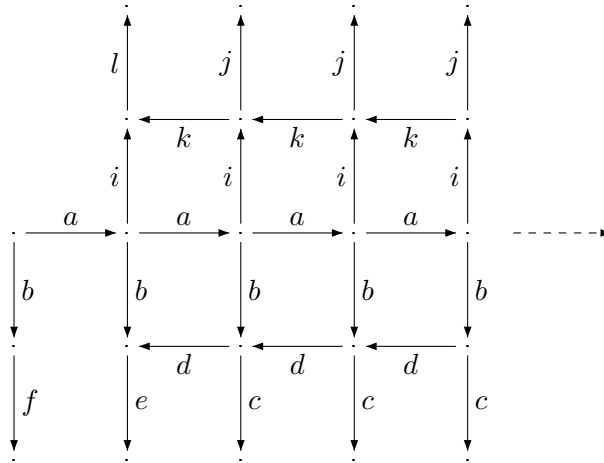
On obtient $K \in \text{Graphe}_3$ et K est le mot de Champernowne.

3.2 Mot de Liouville

Le nombre de Liouville est défini par $l = \sum_{n=1}^{\infty} 10^{-n!} = 0.11000100\dots$. Nous appellerons *mot de Liouville* la partie décimale de ce nombre. Son $i^{\text{ème}}$ caractère est un 1 si i est la factorielle d'un entier.

Le nombre de 0 entre le $n^{\text{ème}}$ et le $n+1^{\text{ème}}$ 1 est égal à : $(n+1)! - n! - 1 = nn! - 1$. Nous allons utiliser cette propriété pour montrer que le mot de Liouville est au niveau 3 de la hiérarchie.

On part du graphe G suivant :



On a $G \in \text{Graphe}_1$. On prend r la racine de G , et on construit le graphe $H = g^{-1}(\text{Dep}(G, r))$ où g est l'application rationnelle suivante :

$$\begin{aligned} a &\mapsto a & ; & & d &\mapsto d & & ; & j &\mapsto j \\ b &\mapsto b & ; & & e &\mapsto \bar{e}\bar{d}^* \bar{b}i & ; & & k &\mapsto k \\ c &\mapsto c & ; & & f &\mapsto f & ; & & l &\mapsto l \end{aligned}$$

suivie de :

$$\begin{aligned}
a &\mapsto a \\
b &\mapsto b \\
c &\mapsto c \\
d &\mapsto dc\bar{c} + de\bar{e}dc \\
e &\mapsto \bar{c}\bar{d}c + \bar{d}\bar{c}\bar{c}de + \bar{b}be \\
f &\mapsto f \\
j &\mapsto j \\
k &\mapsto kj\bar{j} + k\bar{l}\bar{l}\bar{k}j \\
l &\mapsto \bar{j}\bar{k}j + \bar{j}\bar{e}\bar{d}^*\bar{b}\bar{a}bd^*e + \bar{e}\bar{b}\bar{a}b
\end{aligned}$$

Donc $H \in \text{Graphe}_2$ et nous construisons le graphe $K = h^{-1}(\text{Dep}(H, r))$ où h est l'application rationnelle suivante :

$$\begin{aligned}
a &\mapsto a \\
b &\mapsto bc^*e(j+l)^*f + \\
&\quad \bar{f}(\bar{l} + \bar{k})^*\bar{j}k(j+l)^*f + \\
&\quad \bar{f}(\bar{l} + \bar{k})^*\bar{e}(\bar{e} + \bar{d})^*\bar{c}d(c+e)^*(j+l)^*f \\
f &\mapsto \bar{f}(\bar{l} + \bar{k})^*(\bar{e} + \bar{d})^*\bar{b}b(e+d)^*(l+k)^*f\bar{f}
\end{aligned}$$

suivie de :

$$\begin{aligned}
0 &\mapsto \bar{b}bb \\
1 &\mapsto f\bar{f}\bar{b}^*ab
\end{aligned}$$

On obtient $K \in \text{Graphe}_3$ et K est la partie décimale du nombre de Liouville.

3.3 D'autres mots infinis

Nous venons de montrer que deux mots infinis particuliers sont dans la hiérarchie. Mais cette manière d'explorer les différents niveaux est fastidieuse et impose de connaître à l'avance le mot recherché. Il n'est actuellement pas possible de caractériser de manière globale l'ensemble des mots infinis présents à un niveau de la hiérarchie.

Chapitre 4

Arbres algébriques et mots infinis

Nous disposons maintenant d'une hiérarchie de familles de graphes sur lesquelles la logique monadique est décidable. Mais seuls les niveaux 0 et 1 sont bien connus. Nous allons donc ici étudier le niveau 2, qui est peu connu, et pour cela nous nous intéresserons aux mots infinis, qui sont les graphes les plus simples à étudier, mais qui vont poser des bases pour une étude plus complète.

4.1 Grammaires de termes

On a vu qu'au niveau 1 de la hiérarchie, les graphes des automates à pile sont engendrés par les grammaires de graphes. Au niveau 2 de la hiérarchie, on a la famille des arbres algébriques ($tree_2$) qui est obtenue par dépliage des graphes préfixe reconnaissables. On va présenter ici une autre forme de grammaire qui engendre cette famille.

Les graphes au niveau 0 de la hiérarchie sont les graphes finis, les mots de ce niveau sont donc les mots finis. Au niveau 1, les mots que l'on obtient sont des graphes réguliers. Ils sont finiment décomposables par distance, et donc ultimement périodiques.

Définition 4.1 *Soit un alphabet Σ et une relation d'arité $\delta : \Sigma \mapsto \mathbb{N}$. Un terme est un mot de Σ^* de la forme $ft_1 \dots t_{\delta(f)}$ où $f \in \Sigma$ et $t_1, \dots, t_{\delta(f)}$.*

On appellera *terme constant* un terme réduit à un élément de Σ d'arité nulle. Un terme v est un *sous-terme* d'un terme u si v est un facteur de u .

Définition 4.2 *Soient t, t_1, \dots, t_n des termes et x_1, \dots, x_n des termes constants. La substitution $t[x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n]$ définit le terme t dans lequel on a substitué à chaque occurrence de x_1, \dots, x_n le terme t_1, \dots, t_n correspondant.*

Définition 4.3 *Un terme t sur un alphabet Σ peut aussi être vu comme un arbre ordonné, coloré sur Σ :*

- si t est constant, son arbre est réduit à un seul sommet coloré par t ,
- sinon t est de la forme $ft_1 \dots t_{\delta(f)}$ avec $f \in \Sigma$ et $t_1, \dots, t_{\delta(f)}$ des termes et son arbre a pour racine un sommet étiqueté par f , et qui a comme arbre-fils les arbres de $t_1, \dots, t_{\delta(f)}$ dans l'ordre.

Remarque 3 *Par la suite, un terme sera indifféremment vu comme un mot ou un arbre. On ne précisera que si le contexte n'est pas explicite.*

Définition 4.4 *Une grammaire de termes H est un quintuplet $(\Sigma, N, \delta, R, S)$ composé des ensembles suivants :*

- un ensemble de terminaux Σ ,
- un ensemble de non-terminaux N ,
- une relation d'arité $\delta : N \mapsto \mathbb{N}$,
- un ensemble R de règles de la forme $P_{x_1, \dots, x_{\delta(P)}} \rightarrow t$ avec :
 - $P \in N$ un non-terminal d'arité $\delta(P)$,
 - $\chi = \{x_1, \dots, x_{\delta(P)}\}$ un ensemble de variables distinctes et d'arité nulle,
 - t un terme sur l'alphabet $N \cup T \cup \chi$,
- un axiome $S \in N$ d'arité nulle.

Par la suite, pour toute règle $r \in R$, on notera

- $\mathcal{N}(r)$ le non-terminal partie gauche de r ,
- $\mathcal{T}(r)$ le terme partie droite de r ,
- $r|_{\Sigma}$ l'ensemble des terminaux étiquétant $\mathcal{T}(r)$,
- $r|_N$ l'ensemble des non-terminaux étiquétant $\mathcal{T}(r)$.

Définition 4.5 *Pour tout terme $t = Pt_1 \dots t_{\delta(P)}$ où P est un non-terminal, on dit que t est un appel à P et que les termes $t_1, \dots, t_{\delta(P)}$ sont les paramètres de cet appel.*

L'application d'une règle de réécriture $r \in R$ à un terme t consiste à substituer à t , un sous-terme t' de t qui est un appel à $\mathcal{N}(r)$, par le terme $\mathcal{T}(r)[x_1 \leftarrow p_1, \dots, x_n \leftarrow p_n]$ avec p_1, \dots, p_n les paramètres d'appel de t' .

Définition 4.6 *La réécriture parallèle sur un terme t consiste à réécrire tous les non-terminaux de t .*

Il nous reste maintenant à définir l'arbre engendré par une grammaire de terme.

Définition 4.7 *On appellera la cime d'un arbre A représentant un terme, le plus grand sous-arbre de A qui contienne la racine de A et tel que tout sommet étiqueté par un non-terminal soit une feuille.*

On appellera la cime stricte d'un arbre A représentant un terme, la cime de A privée de ses sommets étiquetés par un non-terminal.

Définition 4.8 *On dit qu'un terme t est inclus dans un terme t' si la cîme stricte de l'arbre de t est un sous-arbre de la cîme stricte de l'arbre de t' .*

Propriété 10 *Pour t un terme et t' le terme t obtenu après un pas de réécriture on a $t \subseteq t'$.*

Définition 4.9 *Un arbre A est une extension d'un terme t si la cîme stricte de l'arbre de t est un sous-arbre de A .*

L'union d'une série de termes t_1, \dots, t_n est le plus petit arbre A qui soit une extension de chacun des termes t_1, \dots, t_n .

Définition 4.10 *On définit l'arbre engendré par une grammaire de termes comme étant l'union de la suite croissante éventuellement infinie de termes t_1, t_2, \dots telle que $t_1 = \mathcal{T}(S)$ et pour tout i , t_{i+1} est le terme obtenu par de réécriture parallèle à partir de t_i .*

Afin de s'assurer que l'arbre engendré existe, on utilisera des grammaires en forme normale de tête.

Définition 4.11 *Une grammaire est en forme normale de tête si pour toute règle $r \in R$, $\mathcal{T}(r)$ n'est pas un appel.*

La propriété suivante nous assure que l'on pourra toujours se ramener à une telle grammaire.

Propriété 11 *Toute grammaire de termes peut-être mise en forme normale de tête.*

Preuve : Pour chaque règle r d'une grammaire de termes H , si r n'est pas en forme normale de tête, alors :

- soit $\mathcal{T}(r)$ est une variable, il suffit donc de récrire toutes les occurrences de $\mathcal{N}(r)$ dans la grammaires,
- soit la racine de $\mathcal{T}(r)$ est coloré par un non-terminal, et dans ce cas, soit par réécriture parallèle on se ramène à une règle en forme normale de tête, soit ce n'est pas possible, auquel cas cette règle ne peut engendrer un arbre, on la remplace donc par la règle $\mathcal{N}(r) \rightarrow \varepsilon$.

□

Remarque 4 *La propriété précédente implique que l'on accepte qu'une grammaire de termes qui n'engendre pas d'arbre soit transformée en une grammaire réduite à un axiome $S \rightarrow \varepsilon$.*

Propriété 12 *Les grammaires de termes engendrent exactement les arbres algébriques qui sont des termes.*

Remarque 5 *Par extension, on appellera mot des feuilles d'une grammaire le mot des feuilles de l'arbre engendré par cette grammaire.*

4.2 Forme réduite des grammaires de termes

Étant donné que l'on ne s'intéresse qu'au mot des feuilles de l'arbre engendré par une grammaire de termes, il est possible de simplifier celle-ci, et de la mettre sous une forme qui va nous permettre de dégager plus facilement le résultat que l'on souhaite obtenir.

Exemple 12 Soit la grammaire suivante :

$$\begin{aligned} S &\mapsto kcQab \\ Q &\mapsto fcPx_1x_2Ua \\ U &\mapsto hx_1Ux_1 \\ V &\mapsto ghax_1x_2 \\ P &\mapsto gx_1x_2Ox_1x_2 \\ O &\mapsto Phx_1x_2hx_2x_1 \end{aligned}$$

avec les fonctions suivantes :

$$\begin{aligned} \Sigma &: \{k : 2, f : 3, g : 3, h : 2, a : 0, b : 0, c : 0\} \\ \text{et } N &: \{S : 0, Q : 2, U : 1, V : 2, P : 2, O : 2\} \end{aligned}$$

Elle engendre l'arbre de la figure 4.1. On peut voir sur cet arbre que certains sommets n'interviennent pas dans le mots des feuilles, ils seraient donc intéressants de les supprimer. De même, on peut se douter qu'une grammaire plus simple pourrait engendrer un graphe ayant le même mot des feuilles.

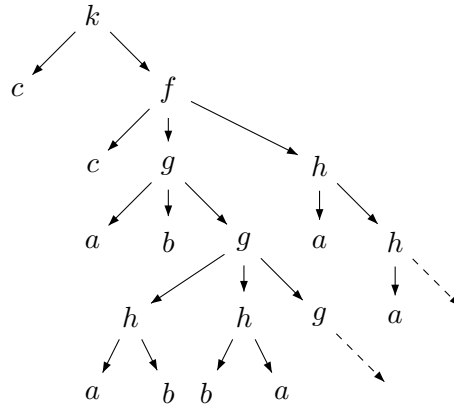


FIG. 4.1 – Graphe engendré par la grammaire de l'exemple 12.

4.2.1 Grammaire de termes développée

La première transformation que l'on va utiliser est la mise sous forme développée d'une grammaire de termes.

On appellera *terme fini* un terme t tel qu'il n'existe pas de sous-terme de t qui soit un appel.

Définition 4.12 Une règle r est une règle finie si par réécriture itérée de $\mathcal{T}(r)$, on obtient un terme fini.

Définition 4.13 Une règle r est une règle utile si il existe une suite de réécriture à partir de l'axiome tel qu'il existe un sous-terme du terme obtenu qui soit un appel à r .

Définition 4.14 Soit une règle r , une variable x_i de r est une variable utile si il existe dans $\mathcal{T}(r)$ un sous-terme x_i qui ne soit pas un sous-terme du i ème paramètre d'un appel à $\mathcal{N}(r)$.

Ce qui nous permet de définir ce qu'est une grammaire sous forme développée.

Définition 4.15 Une grammaire est sous forme développée si pour toute règle $r \in R$, r n'est pas finie, r est utile et toutes les variables de r sont utiles.

Il est possible de mettre toute grammaire de termes sous forme développée :

1. pour chaque règle finie r , on remplace r par la règle récrivant $\mathcal{N}(r)$ en le terme fini que l'on obtient par réécriture de $\mathcal{T}(r)$,
2. pour chaque règle r , si r contient un sous-arbre qui est un appel à une règle finie, on réécrit ce sous-arbre,
3. on supprime toutes les règles et tous les paramètres qui ne sont pas utiles.

Les deux premières étapes rendent les règles finies inutiles ; elles sont donc supprimées lors de la troisième étape. La grammaire obtenue est bien sous forme développée et préserve le mot engendré.

Propriété 13 Mettre une grammaire sous forme développée ne change pas le mot qu'elle engendre.

Preuve : Les deux premières étapes ne sont que des applications de réécritures, elles ne modifient donc pas l'arbre engendré. La troisième étape supprime des règles et des paramètres qui, par définition, n'ont aucune influence sur l'arbre engendré.

La mise sous forme développée ne modifie donc pas l'arbre engendré par une grammaire de termes. \square

Exemple 13 *La grammaire de l'exemple 12 mise sous forme développée est la grammaire :*

$$\begin{aligned} S &\mapsto kcQab \\ Q &\mapsto fcPx_1x_2Ua \\ U &\mapsto hx_1Ux_1 \\ P &\mapsto gx_1x_2Ox_1x_2 \\ O &\mapsto Phx_1x_2hx_2x_1 \end{aligned}$$

La règle V étant inutile, elle est supprimée.

4.2.2 Grammaire de terme élaguée

On va maintenant définir une deuxième transformation, qui cette fois ne conserve pas l'arbre engendré, mais qui préserve le mot des feuilles.

Définition 4.16 *Un terme t est dit élagué si, la cîme de t contient au maximum un sommet coloré par un non-terminal, que ce non-terminal est le sommet le plus à droite et que chacun de ses paramètres est un terme élagué.*

Définition 4.17 *Une grammaire développée est dite élaguée si pour toute règle $r \in R$ $\mathcal{T}(r)$ est un terme élagué.*

Il est possible de transformer une grammaire de termes, en une grammaire de termes élaguée de la manière suivante : Soit $t = ft_1 \dots t_{\delta(f)}$ un terme, on élague t de la manière suivante :

- si aucun des paramètres $t_1 \dots t_{\delta(f)}$ ne possède de sous-terme qui soit un appel, alors t est élagué,
- si f est un non-terminal, on élague ses paramètres,
- sinon, soit t_i le premier terme contenant un sous-terme qui soit un appel, on élague les termes $t_1 \dots t_i$ puis on fixe l'arité de f à i .

Cette procédure est assurée de terminer, car la taille des termes que l'on élague ne peut que diminuer. On élague une grammaire développée en élaguant chacun des termes apparaissant en partie droite de ses règles.

Exemple 14 *La grammaire de l'exemple 13 mise sous forme élaguée est la grammaire :*

$$\begin{aligned} S &\mapsto kcQab \\ Q &\mapsto fcPx_1x_2 \\ P &\mapsto gx_1x_2Ox_1x_2 \\ O &\mapsto Phx_1x_2hx_2x_1 \end{aligned}$$

On note que f à maintenant une arité égale à 2 et que la règle U a été supprimée.

La forme élaguée d'une grammaire de termes est intéressante car la propriété 1 nous donne la propriété :

Propriété 14 *Le mot des feuilles d'une grammaire de termes est conservé si l'on élague cette grammaire.*

Une grammaire sous forme élaguée est telle que pour toute règle $r \in R$, la cime du terme $\mathcal{T}(r)$ et de tous ses sous termes contient au maximum un sommet étiqueté par un non-terminal, et ce sommet s'il existe est la feuille la plus à gauche.

Définition 4.18 *Une réécriture stable d'une règle r d'une grammaire de terme élaguée, consiste à récrire le terme $\mathcal{T}(r)$, puis à élaguer la grammaire obtenue.*

De par sa définition, la réécriture stable d'une grammaire de termes élaguée donne une grammaire de termes élaguée dont l'arbre engendré à le même mot des feuilles que l'arbre engendré de la première grammaire.

4.2.3 Grammaire de terme récursive simple

On appellera *appel principal* d'un terme t élagué, le sous-terme t' de t s'il existe, tel qu'il n'existe pas de sous-terme de t qui soit un appel et dont t' est un sous-terme.

Définition 4.19 *Une règle $r \in R$ d'une grammaire de termes élaguée est récursive directe si l'appel principal de $\mathcal{T}(r)$ est un appel à $\mathcal{N}(r)$.*

Définition 4.20 *Une règle $r \in R$ d'une grammaire de terme élaguée est récursive si par réécriture stable de son appel principal, il est possible d'obtenir une règle récursive directe.*

On peut maintenant définir une nouvelle classe de grammaires.

Définition 4.21 *Une grammaire de termes H est récursive simple si toutes ses règles récursives sont récursives directes et qu'aucun des paramètres d'un appel récursif ne possède de sous-terme qui soit un appel.*

Toute grammaire peut-être transformée en une grammaire équivalente pour le mot des feuilles.

Propriété 15 *Toute grammaire de termes peut-être transformée en une grammaire récursive simple qui engendre un arbre ayant le même mot des feuilles que l'arbre engendré par la grammaire de départ.*

Preuve : Soit H une grammaire de termes élaguée :

1. on transforme chaque règle récursive en une règle récursive directe par réécriture stable de son appel principal,

2. pour chaque règle récursive $r \in R$ si un des paramètres de l'appel principal de $\mathcal{J}(r)$ possède un sous-terme qui est un appel, on effectue une réécriture stable de l'appel principal de r ,
3. si au moins une règle a été modifiée on recommence à l'étape 1.

H étant élaguée, tous les paramètres d'un appel sont donc utiles, notamment ceux contenant un sous-terme qui est un appel. À l'étape 2, le pas de réécriture stable va donc faire disparaître l'appel récursif et réduire le nombre d'appels imbriqués dans cette règle.

Étant donné que la procédure se termine quand il n'y a plus d'appel imbriqué et qu'à chaque passage le nombre d'appels imbriqués décroît, la procédure termine en un temps fini.

La seule opération utilisée est la réécriture stable. La grammaire obtenue engendre donc un arbre qui a le même mot des feuilles que l'arbre engendré par la grammaire de départ.

Lorsque la procédure se termine, toutes les règles récursives sont récursives directes et ne contiennent pas d'appel imbriqué, la grammaire H est donc sous forme récursive simple. \square

Exemple 15 *La grammaire de l'exemple 14 mise sous forme récursive simple est la grammaire :*

$$\begin{aligned} S &\mapsto kcQab \\ Q &\mapsto fcPx_1x_2 \\ P &\mapsto gx_1x_2Phx_1x_2hx_2x_1 \end{aligned}$$

4.2.4 Forme réduite

On arrive maintenant à la forme qui va nous intéresser.

Définition 4.22 *Une grammaire de termes est sous forme réduite si elle est récursive simple et qu'elle n'est composée, au maximum, que de deux règles s et p qui ne contiennent pas d'appels imbriqués et telles que $s|_N \subseteq \{p\}$ et $p|_N \subseteq \{p\}$.*

Propriété 16 *Toute grammaire de termes peut-être transformée en une grammaire réduite qui engendre un arbre ayant le même mot des feuilles que l'arbre engendré par la grammaire de départ.*

Preuve : Soit H une grammaire sous forme récursive simple. Soit l'axiome est une règle récursive, alors H est réduite à l'axiome et est sous forme réduite. sinon

1. par réécriture stable itérée de l'appel principal de l'axiome, on se ramène à un axiome dont l'appel principal est un appel à une règle récursive,
2. si l'axiome ne contient pas d'appels imbriqués, H est sous forme réduite, sinon on applique un pas de réécriture stable et on recommence la procédure en 1.

H étant sous forme récursive simple et le nombre de règles étant fini, l'étape 1 se fait en un nombre fini de récritures. La procédure se termine lorsque l'axiome ne contient plus d'appels imbriqués, or l'étape 2 nous assure que le nombre d'appel imbriqué décroît, la procédure se termine donc en un nombre fini d'étapes.

La seule opération utilisée est la récriture stable, le mot des feuilles de l'arbre engendré ne change donc pas, la grammaire H est donc bien sous forme réduite. \square

Exemple 16 La grammaire de l'exemple 15 mise sous forme réduite est la grammaire :

$$\begin{aligned} S &\mapsto kcf cPab \\ P &\mapsto gx_1x_2Phx_1x_2hx_2x_1 \end{aligned}$$

Elle engendre le graphe de la figure 4.2.

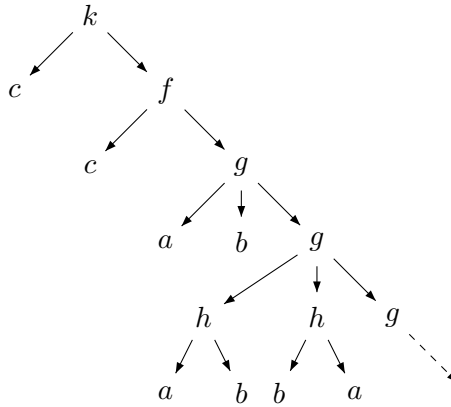


FIG. 4.2 – Graphe engendré par la grammaire de l'exemple 16.

4.3 Mots infinis au deuxième niveau

4.3.1 Arbres algébriques

On dispose maintenant d'une forme simplifiée pour les grammaires de termes qui va nous permettre d'énoncer les deux lemmes ci-dessous :

Lemme 1 *Les mots des feuilles des arbres algébriques sont des mots morphiques.*

Preuve : Soit H une grammaire de termes réduite. Si H engendre un arbre fini, le mot des feuilles de cet arbre est un mot fini qui est donc trivialement morphique.

Soit u l'application qui à un terme fait correspondre son mot des feuilles, et u' l'application qui à un terme t fait correspondre le mot des feuilles de la cîme stricte de t . Soit v l'application qui à un couple (r, i) fait correspondre le mot des feuilles du i ème paramètre de l'appel principal de la règle r .

Le mot des feuilles de H est le mot infini résultat de la concaténation du mot $u'(\mathcal{T}(s))$ et du mot $u(\mathcal{T}(p))$ dans lequel on a substitué à chaque caractère x_i le mot $v(s, i)$.

Or on obtient le même mot en appliquant le morphisme $\sigma : \forall 1 \leq i \leq \delta(p), x_i \mapsto v(s, i)$ et $\Delta \mapsto u'(\mathcal{T}(s))$ au mot $\Delta u(\mathcal{T}(p))$.

Pour t un terme, on définit $u'_n(t)$ comme étant l'application de u' au terme t après n pas de réécriture stable. Le mot infini $u(\mathcal{T}(p))$ est égal à $u'_\omega(\mathcal{T}(p))$.

Soit τ le morphisme : $\forall 1 \leq i \leq \delta(p), x_i \mapsto v(p, i)$. Le mot $u'_n(\mathcal{T}(p))$ est la concaténation pour j allant de 0 à n des $\tau^j(u'(\mathcal{T}(p)))$. Pour $n = \omega$, on étend τ avec $\Delta \mapsto \Delta u'(\mathcal{T}(p))$ et l'on obtient $\tau^\omega(\Delta) = \Delta u(\mathcal{T}(p))$.

Le mot des feuilles de H est donc le mot $\sigma(\tau^\omega(\Delta))$, et est un mot morphique. \square

Exemple 17 La grammaire de l'exemple 16 à pour mot des feuilles le mot morphique défini par les morphismes suivants :

$$\begin{aligned}\sigma : \Delta &\mapsto cc, x_1 \mapsto a, x_2 \mapsto b \\ \tau : \Delta &\mapsto \Delta x_1 x_2, x_1 \mapsto x_1 x_2, x_2 \mapsto x_2 x_1\end{aligned}$$

Le mot $\sigma(\tau^\omega(\Delta))$ est le mot $ccababba\dots$

Lemme 2 Les mots morphiques sont des mots des feuilles des arbres algébriques.

Preuve : Soit Σ un alphabet et $\sigma(\tau^\omega(a))$ un mot morphique m sur cet alphabet. σ (resp. τ) est un morphisme $x_i \mapsto u_i$ (resp. $y_j \mapsto v_j$, avec $y_1 = a$ et $v_1 = av'_1$) de Σ dans Σ^* .

On défini H une grammaire de termes. H à pour terminaux, les éléments de Σ qui seront d'arité nulle et des éléments s_i et t_i qui seront d'arité respective $|u_i|$ et $|v_i|$, ainsi que s_0 d'arité $|\sigma(a)| + 1$. L'ensemble des terminaux de H est réduit à deux éléments : S d'arité nulle (l'axiome) et P d'arité égale à $\delta(\tau)$:

$$\begin{aligned}- R &\mapsto s_0 \sigma(a) P s_1 u_1 \dots s_{\delta(\sigma)} u_{\delta(\sigma)} \\ - P &\mapsto t_1 v'_1 P t_1 v_1 \dots t_{\delta(\tau)} v_{\delta(\tau)}\end{aligned}$$

La grammaire H à m pour mot des feuilles. \square

Ces deux lemmes nous permettent maintenant d'énoncer le théorème suivant :

Théorème 4.1 Les mots des feuilles des arbres algébriques sont les mots morphiques.

On dispose de plus de la propriété suivante :

Propriété 17 ([Cau02]) *Les mots des feuilles des arbres algébriques sont dans Graphe2.*

Qui nous permet de réobtenir ce résultat déjà connu :

Corollaire 1 *Les mots morphiques sont au niveau 2 de la hiérarchie.*

4.4 Perspectives

Nous venons de voir qu'au niveau 2 de la hiérarchie se trouve les mots morphiques. Mais existe-t-il d'autres mots infinis à ce niveau ? On conjecture que non : les mots infinis du niveau 2 seraient exactement les mots morphiques, mais la preuve reste à terminer.

Chapitre 5

Transducteurs et mots infinis

Une autre manière d'obtenir des graphes infinis est d'utiliser les transducteurs. On va donc ici étudier les mots infinis engendrés par des transducteurs et voir quels sont les liens qui les unissent au mots infinis présents dans la hiérarchie.

5.1 Transducteurs strictement croissants

Nous allons commencer par nous restreindre à une famille de transducteurs simples, et voir quels sont les mots infinis obtenus.

Définition 5.1 *On appellera transducteur croissant (resp. strictement croissant) un transducteur tel que pour tout chemin $p \xrightarrow{u/v} q$, on a $|u| \leq |v|$ (resp. $|u| < |v|$).*

Théorème 5.1 *Les mots infinis engendrés par les transducteurs strictement croissants sont les mots ultimement périodiques.*

Preuve : Soit T un transducteur strictement croissant engendrant un mot m infini. Pour chaque symbole s de l'alphabet de sortie, on considère l'automate A construit à partir de T :

- $Q_A = Q_T$,
- $\Gamma_A = \{a\}$,
- $G_A = \{(p, a, q) \mid \exists u, v \ (p, u, v, q) \in G_T\}$,
- $I_A = I_T$,
- $F_A = \{x \mid x \in F_T \wedge L_T(x) = s\}$.

Ces automates engendrent des mots ultimement périodiques, m est donc une union finie de mots ultimement périodiques, et donc m est, lui aussi, un mot ultimement périodique dont la période est le plus petit multiple commun des périodes des mots dont il est l'union. \square

5.2 Perspectives

Nous venons de montrer que les mots infinis engendrés par les transducteurs strictement croissants, sont les mots ultimement périodiques, c'est-à-dire les mots que l'on retrouve au niveau 1 de la hiérarchie.

Qu'en est-il pour les transducteurs croissants? On conjecture qu'ils engendrent les mots morphiques, c'est-à-dire les mots que l'on pense être au niveau 2 de la hiérarchie.

Il serait donc intéressant de continuer d'étudier ces deux familles de graphes ainsi que les relations qu'il peut y avoir entre elles.

Conclusion

Lors de ce stage, deux voies majeures ont été étudiées en parallèle, à savoir, les graphes de niveau 2, ainsi que les graphes reconnus par les automates synchronisés.

Des premiers résultats sur les mots infinis présents au niveau 2 ont été apportés, mais de nombreuses pistes restent encore à être étudiées. Est-ce que les mots des feuilles des arbres algébriques sont exactement les mots présent dans les graphes du deuxième niveau ?

Si cette conjecture se révèle être vraie, les mots du deuxième niveau seraient exactement les mots morphiques. Il serait ensuite possible d'essayer de généraliser cette conjecture à tous les niveaux : les mots des feuilles de $Arbre_n$ sont les mots de $Graphe_n$, ce qui permettrait de définir les mots morphiques d'ordre supérieur.

Il reste aussi à terminer les recherches entamées sur les mots infinis que l'on trouve, parmi les graphes obtenus à l'aide de transducteurs. Mais aussi sur les liens qui existent entre ces graphes et les graphes de la hiérarchie.

On peut donc voir que même si des premiers résultats ont été apportés, il reste bien des possibilités de recherche.

De plus, l'étude des mots infinis n'est qu'un premier pas dans la compréhension des familles des graphes du niveau 2 (et plus) de la hiérarchie.

Bibliographie

- [Bü60] R. Büchi. On a decision method in restricted second order arithmetic. *Proc. Int. Congress on Logic, Methodology and Philosophy of Science*, pages 1–11, 1960.
- [Cau95] D. Caucal. *Bisimulation of context-free grammars and of pushdown automata*, volume 53 of *Modal logic and process algebra*. CSLI Lecture Notes, Stanford, 1995.
- [Cau96] D. Caucal. On infinite transition graphs having a decidable monadic theory. In *ICALP'96*, volume 1099 of *LNCS*, pages 194–205, 1996.
- [Cau02] D. Caucal. On infinite terms having a decidable monadic theory. In *MFCS'02*, pages 79–115. Springer Verlag, 2002.
- [Cou90] B. Courcelle. Graph rewriting : an algebraic and logic approach. *Handbook of TCS*, B :193–242, 1990.
- [CW98] B. Courcelle and I. Walukiewicz. Monadic second-order logic, graph coverings and unfoldings of transition systems. In *Annals of Pure and Applied Logic*, 1998.
- [Har78] Michael A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley Publishing Company, 1978.
- [HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, 1979.
- [Kle56] S. C. Kleene. Representation of events in nerve nets and finite automata. In *C.E. Shannon and J. McCarthy, Annals of Mathematics Studies*, pages 3–41, Princeton, 1956. Princeton University Press.
- [MS85] D. Muller and P. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37 :51–75, 1985.
- [Pri00] Christophe Prieur. *Fonctions rationnelles de mots infinis et continuité*. PhD thesis, LIAFA - Paris VII, 2000.
- [Rab69] M. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. soc.*, 141 :1–35, 1969.