Synchronisation de grammaires de graphe

Stéphane Hassen

Institut de Recherche en Mathématiques et Informatique Appliquées
Université de la Réunion

Thèse en vue de l'obtention du titre de
docteur de l'université de la Réunion
en informatique

Synchronisation de grammaires de graphes

Stéphane Hassen

Composition du jury :

Olivier Carton (rapporteur)
Didier Caucal (directeur)
Jean Diatta (co-directeur)
Damian Niwinski (rapporteur)
Henry Ralambondrainy (président)

Remerciements

Mes remerciements vont tout d'abord à messieurs Carton et Niwinsky, pour leur temps et l'implication dont ils ont fait preuve pour les rapports sur cette thèse. Je ne pense pas pouvoir retranscrire ici la gratitude que j'éprouve pour Didier Caucal pour avoir accepté la responsabilité de directeur ainsi que pour m'avoir soutenu durant toutes ces années. Merci également à Jean Diatta pour m'avoir permis de retrouver la légalité vis-à-vis de l'université en acceptant de devenir co-directeur de cette thèse. Merci également à Theodor Knapik pour avoir commençé à m'encadrer et surtout m'avoir donné envie d'effectuer cette thèse.

Mes pensées vont également aux membres des laboratoires dont j'ai arpenté les couloirs, à savoir l'IREMIA, l'IRISA de Rennes et l'IGM de Marne-La-Vallée. Bien evidemment je ne pourrais pas citer tous les noms mais pendant que j'écris ces lignes, vos visages me reviennent et c'est vers vous que se dirigent mes remerciements pour l'acceuil que j'ai reçu, les discussions scientifiques aussi bien que philosophiques auxquelles j'ai pu participer...

Last but not least, merci à ma famille, celle du sang et celle du coeur, qui même si elle n'a jamais vraiment compris mon travail n'a jamais douté de moi et m'a supporté pendant toutes ces années.

Table des matières

Chapitre 1

Introduction

L'étude des automates commence avec l'article d'Alan Turing [Tu36] et sa description d'un modèle de calcul : les machines (ou automates) de Turing. Ces machines théoriques sont constituées d'une bande infinie, d'une tête de lecture/écriture (ayant un nombre fini d'états possibles) pouvant parcourir la bande et d'un programme dictant à la machine son fonctionnement : les actions de la tête dépendent de son état et de la lettre lue sur la bande. Ces modèles permettent de définir des langages : un mot est accepté par un automate si l'application de son programme sur une bande contenant (uniquement) ce mot amène la tête de lecture dans un état final prédéfini. L'ensemble des mots acceptés par une machine forme ainsi le langage accepté par la machine. En restreignant ce modèle de sorte que l'écriture ne se soit pas possible et que la tête ne se déplace que dans un sens, on obtient le modèle théorique des automates finis. Les langages acceptés par ces machines restreintes sont exactement les langages clos par union, intersection et itération : ce sont les langages réguliers.
En 1956, Stephen Kleene montre que les automates finis reconnaissent exactement les langages réguliers [Kl56]. Ce résultat, appelé le théorème de Kleene, est obtenu en considérant d'une part la description d'une machine par ses composants locaux, un réseau fini de neurones de McCulloch et Pitts [MP43], mais aussi en définissant une machine par un nombre fini d'états globaux.
En se basant sur la description formelle des automates finis, Rabin et Scott

[RS59] reformulent le théorème de Kleene en termes algébriques. Les langages acceptés par les automates finis forment une algèbre de Boole qui est de plus close par concaténation et itération de la concaténation. Il en résulte que l'inclusion des langages réguliers est décidable. Le lecteur pourra se reporter à la synthèse de Dominique Perrin sur l'histoire des automates finis [Pe95].
Dans ce document, nous représentons les automates par des relations binaires, décrivant les transitions du programme de la machine, entre un nombre fini d'éléments. Cette similarité entre automates et structures algébriques a été mise en exergue par Marcel-Paul Schützenberger [Sc55]. Une représentation naturelle de ces structures se fait par graphes finis : les sommets sont les états de l'automate et les arcs étiquetés expriment les relations de transition. Le langage accepté par un automate est alors donné par les traces du graphe le représentant : les langages réguliers sont les traces des graphes finis.
En terme de complexité, on peut classer les langages selon la hiérarchie de Chomsky [Ch56]. Dans sa recherche d'une grammaire de mots pour exprimer la richesse de la langue anglaise, Noam Chomsky dégage la notion de grammaire algébrique. Les grammaires algébriques engendrent les langages dits algébriques incluant les langages réguliers.
Tout comme les automates finis reconnaissent les langages régulier, les automates le plus souvent associés aux langages algébriques sont les automates à pile. Ces machines sont une restriction des machines de Turing : la tête ne peut que lire et est uni-directionnelle. En 1985, David Müller et Paul Schupp [MS85] ont étudié la structure des graphes de fonctionnement de ces automates : les sommets sont les configurations de la machine et ils sont reliés par des arcs décrivant ses actions. Ils montrent que ces graphes de transition coïncident avec la classe des graphes de degré fini qui sont finiment décomposables par distance. Ils donnent ainsi une description de la structure des graphes de transition des automates à pile.
On a alors cherché à construire ces graphes par l'utilisation de grammaires, non plus de mots mais de graphes associant à un arc, dit non-terminal, un unique graphe fini ([Co89]). Par récritures itérées à partir d'un graphe fini, on engendre un graphe dit régulier. Ces graphes sont une généralisation des graphes des automates à pile : les graphes réguliers de degré fini sont les

graphes de transition des automates à pile.
Néanmoins, les traces des graphes réguliers sont exactement les langages algébriques. La représentation externe des graphes réguliers par ces grammaires fournit alors un nouvel outil pour l'étude des langages algébriques. Nos lecteurs pourront trouver dans [Ca07] une synthèse sur ces grammaires de graphes.
Contrairement aux langages réguliers, les langages algébriques ne forment pas une algèbre de Boole, notamment l'intersection de deux langages algébriques n'est pas nécessairement un langage algébrique. De plus, par réduction du problème de correspondance de Post, on déduit que l'inclusion des langages algébriques n'est pas décidable. Ceci reste valable pour de nombreux sous-ensembles stricts de ces langages. Citons notamment les langages déterministes « finite-turn » [Va74], les algébriques déterministes à un compteur [VP75] ou les langages simples [Fr76]. Indiquons que c'est pour les langages simples qu'a été donné un premier algorithme d'équivalence des langages algébriques déterministes [KH66].
Néanmoins on peut définir des sous-ensembles de langages algébriques déterministes partageant les propriétés de clôture des langages réguliers. Dans [AM04], Rajeev Alur et Parthasarathy Madhusudan ont défini une restriction des automates à pile, appelés visible, qui sont une simple extension des automates «input-driven» [Mel80] : le fonctionnement de l'automate ne dépend que des caractères d'entrée de la bande de lecture. Outre le fait que les langages (algébriques) visibles, acceptés par ces automates, soient déterministes, ils forment une algèbre de Boole close par concaténation et étoile de Kleene étendant ainsi les propriétés de clôture de base des langages réguliers. D'autres travaux ont alors étendu ces résultats définissant des ensembles plus larges (par inclusion) de langages algébriques ayant ces propriétés. Dans [Ca06], Caucal définit par transducteur fini une relation, dite de synchronisation, entre les graphes réguliers engendrés par des grammaires de graphes synchronisées. Il montre alors que pour un transducteur donné, l'ensemble des traces des graphes réguliers synchronisés forme une algèbre de Boole.
Citons également dans ce domaine les travaux sur les automates à hauteur de pile déterministe par Dirk Nowotka et Jiri Srba [NS07]. Un automate à

pile est à hauteur de pile déterministe si après chaque lecture d'un mot la hauteur de la pile est la même. Partant des travaux de Caucal, Nowotka et Srba définissent une relation de synchronisation entre automates à pile basée sur la hauteur de pile : deux automates sont synchronisés si pour tout mot lu, les hauteurs de pile possibles sont identiques. Cette relation est une relation d'équivalence sur les automates à pile. Ils montrent alors que l'ensemble des langages algébriques acceptés par les automates à hauteur de pile déterministe d'une même classe d'équivalence est clos par union et intersection. De plus, pour les cas où ces automates sont temps-réels et déterministes, cet ensemble de langages est clos par complémentaire.

Le travail présenté ici vise à étendre ces résultats. Nous avons cherché un nouveau moyen de définir des sous-ensembles de langages algébriques contenant les langages réguliers et formant des algèbres de Boole. Pour cela, nous définissons une relation de synchronisation entre grammaires de graphes. A partir d'une grammaire, on définit un poids sur les sommets du graphe engendré, ce que l'on pourrait assimiler à la hauteur de pile chez [NS07]. Ainsi, une grammaire de référence synchronise une autre grammaire si tout mot étiquetant un chemin initial dans le graphe engendré par cette autre grammaire étiquette également un chemin initial dans le graphe référent (*i.e.* le graphe engendré par la grammaire référente) et que ces chemins terminent en des sommets de même poids. Dans cette étude, nous considérons une classe de grammaires particulières : les grammaires pondérées. Ces grammaires engendrent des graphes possiblement non-déterministes mais vérifient la propriété suivante : si il existe deux chemins de même étiquette partant d'un même sommet alors les buts de ces chemins ont le même poids. Nous étudions alors les ensembles de langages algébriques composés de ceux engendrés par les grammaires synchronisés par une grammaire de référence donnée. Nous montrons que cet ensemble de langages synchronisés contient les langages réguliers (inclus dans le langage de référence) et forme une algèbre de Boole. Les constructions mises en œuvre dans l'obtention de ce résultat utilisent une généralisation du produit de synchronisation qui est une opération classique sur les graphes finis. De plus, nous montrons que les graphes engendrés par des grammaires pondérées sont déterminisables et que de fait, les langages

synchronisés forment une sous-classe des langages algébriques déterministes. Nous montrons que dans le cas général l'ensemble des langages synchronisés par une grammaire n'est pas clos par concaténation. Néanmoins, nous donnons une condition suffisante sur la grammaire synchronisante pour que cet ensemble soit clos non seulement par concaténation mais aussi par étoile de Kleene.
Les algèbres de Boole définies par synchronisation sont des sous-ensembles de langages algébriques déterministes. On peut généraliser ces résultats en considérant des langages plus généraux. Un langage algébrique est non-ambigu s'il est engendré par une grammaire algébrique admettant une unique dérivation gauche pour tout mot accepté [HU79]. En ramenant ce principe au niveau des graphes, on définit les grammaires de graphes non-ambiguës dont les graphes engendrés acceptent ces langages. Par synchronisation sur ces grammaires, on peut définir des algèbres de Boole de langages non-ambigus [Ca08].

Plan

Chapitre 2 : Préliminaires
Dans ce chapitre, nous définissons les notations de base et nous décrivons les concepts de mots et de graphes. On trouvera notamment dans ce chapitre les définitions de mots et de langages ou encore la définition des graphes (coloriés, étiquetés,...) et de notions s'y rapportant telles que les chemins, les traces, le degré, *etc.*

Chapitre 3 : Des automates finis
Nous commencerons ici par donner des objets finis permettant de décrire des objets infinis. Le premier paradigme défini ici est la grammaire de mots qui permet d'engendrer des langages. Nous décrirons différentes grammaires avant de situer ce travail dans la hiérarchie de Chomsky. Nous nous intéresserons alors à une classe particulière de langages : les langages réguliers. Pour ceux-ci, nous donnons deux manières finies de les construire (ou de les accepter). En premier lieu, on décrit les grammaires de mots engendrant ces langages, puis nous donnons un moyen structurel de les reconnaître : les au-

tomates finis. Enfin, nous définissons les propriétés algébriques des langages réguliers que nous souhaitons étendre avec le travail présenté dans ce document.

Chapitre 4 : Aux automates réguliers
Dans ce chapitre, nous présentons des systèmes finis permettant de travailler avec les langages algébriques. En premier lieu, nous rappelons une forme particulière de systèmes de récriture de mots, à savoir les automates à pile qui sont des machines munies d'une mémoire et permettant suivant un programme donné de construire (ou reconnaître) les langages algébriques. Utilisant la représentation structurelle des transitions de ces automates, on introduit la notion de grammaires de graphes qui est une extension directe de celle des grammaires de mots : on ne récrit plus des lettres mais des relations. C'est sur ces grammaires que nous travaillons pour définir la relation de synchronisation.

Chapitre 5 : Synchronisation de grammaires
Dans ce chapitre, basé sur [CH08], nous définissons la synchronisation de grammaires de graphes. Nous commençons par définir les grammaires pondérées sur lesquelles portent notre étude. Puis nous définissons la relation de synchronisation sur ces grammaires. En utilisant des généralisations du produit de synchronisation de grammaires, nous montrons alors que les grammaires pondérées sont déterminisables. Nous pouvons alors construire des algèbres de Boole de langages algébriques déterministes et nous donnons de plus une condition suffisante sur les grammaires pour que ces algèbres soient closes par concaténation et étoile de Kleene.

Chapitre 6 : Une extension
Dans cette dernière partie, basée sur [Ca08], nous étendons les résultats du chapitre précédent. En considérant des grammaires dites non-ambiguës, on construit par synchronisation des algèbres de Boole de langages algébriques non-ambigus qui sont les langages algébriques tels que pour tout mot, il n'existe qu'un arbre de dérivation le définissant.

Chapitre 2

Préliminaires

Dans ce nous fixons quelques notations de base notamment des notations ensemblistes utilisées dans le reste du document. Nous décrivons ensuite deux structures élémentaires autour desquelles se construisent nos propos : les mots et les graphes.

2.1 Notations

Pour tout ensemble E, on note $2^E = \{P \mid P \subseteq E\}$ l'ensemble des parties de E. Si E est fini, on note $|E|$ son cardinal, et ω désigne le cardinal d'un ensemble dénombrable. Pour tout ensemble E d'ensembles, l'union est notée

$$\bigcup E = \{e \mid \exists P \in E,\ e \in P\}$$

au lieu de $\bigcup_{P \in E} P$. On utilise l'abréviation $[n] = \{1, \ldots, n\}$ pour désigner les n premiers entiers naturels non nuls.
Soit $u = (a_1, \ldots, a_n)$ un n-uplet, on désigne par $\pi_i(u)$ sa $i^{\text{ème}}$ composante :

$$\pi_i(u) = a_i \ .$$

Soit un ensemble E de symboles munis d'une *arité* $\rho : E \to \mathbb{N}$. On note E_n l'ensemble des symboles de E d'arité n : $E_n := \{a \in E \mid \rho(a) = n\}$. Les symboles de E_0 sont appelés des *constantes*.

2.2 Mots

La première structure de base que nous décrivons, et qui est la plus simple, est celle du mot.
On décrit un mot à l'aide d'un ensemble fini non vide Σ de symboles. Ces symboles sont appelés *lettres* et constituent l'*alphabet* Σ. Soit Σ^* l'ensemble des suites finies de lettres de Σ, *i.e.*

$$\Sigma^* \ = \ \{(a_1, \ldots, a_n) \mid n \geq 0 \ \wedge \ a_1, \ldots, a_n \in \Sigma\}.$$

On munit Σ^* d'une opération binaire interne, la *concaténation*, notée « . » :

$$(a_1, \ldots, a_n).(b_1, \ldots, b_m) \ = \ (a_1, \ldots, a_n, b_1, \ldots, b_m).$$

Tout élément $u = (a_1, \ldots, a_n)$ de Σ^* est un *mot* de *longueur* $|u| = n$. On représente ce mot par $a_1 \ldots a_n$ en identifiant (a) par a et en omettant la concaténation. Le mot vide (), de longueur 0, est désigné par ε. Ainsi $(\Sigma^*, ., \varepsilon)$ est le monoïde libre engendré par Σ.
Tout mot u admet une décomposition de la forme $u = xyz$ avec $x, y, z \in \Sigma^*$ et on appelle x un *préfixe* de u et z un *suffixe* de u.

Pour tout mot $u \in \Sigma^*$ et pour tout $1 \leq i \leq |u|$, $u(i)$ désigne la $i^{\text{ème}}$ lettre de u : $u \ = \ u(1) \ldots u(|u|)$. Pour toute lettre $a \in \Sigma$, on note $|u|_a$ le nombre d'*occurrences* de a dans u : $|u|_a \ = \ |\{i \in [|u|] \mid u(i) = a\}|$ et l'ensemble $Occ(u)$ des lettres d'un mot u est donné par :

$$Occ(u) \ = \ \{a \in \Sigma \mid |u|_a \neq 0\} \ = \ \{u(1), \ldots, u(|u|)\}.$$

On appelle un *langage* toute partie de Σ^*.

Exemple 2.2.1. Considérant l'alphabet latin, *these* est un mot de 5 lettres et $these(2) = h$. Le mot t (respectivement se) est un préfixe (respectivement suffixe) de *these*. De plus $Occ(these) = \{t, h, e, s\}$ et $|these|_e = 2$.

En terme de structure, les mots sont des objets linéaires : une lettre a au plus un successeur. Une extension naturelle consiste à définir des objets à structure arborescente. Ces objets, les termes, ne sont pas considérés dans ce document mais nous étudions une généralisation des termes : les graphes.

2.3 Relations binaires

Une *relation binaire* R d'un ensemble E dans un ensemble F est une partie de $E \times F$. On note aussi eRf pour $(e, f) \in R$.
Le *domaine* $Dom(R)$ et l'*image* $Im(R)$ de R sont donnés par

$$Dom(R) = \{e \in E \mid \exists f \in F,\ eRf\} \text{ et } Im(R) = \{f \in F \mid \exists e \in E,\ eRf\}.$$

L'image d'une partie A de E est $R(A) = \{f \in F \mid \exists e \in A,\ eRf\}$. En particulier, on a $Im(R) = R(E)$.
Pour $E = F$, on dit que $R \subseteq E \times E$ est une relation binaire sur E. Dans ce cas,

- R est *réflexive* si eRe pour tout $e \in E$;
- R est *symétrique* si $R^{-1} = R$;
- R est *anti-symétrique* si pour tout eRf et fRe, on a $e = f$;
- R est *transitive* si pour tout eRf et fRg on a eRg.

2.4 Graphes

On se restreint aux graphes dits *simples* (c'est-à-dire sans multiplicité) et *orientés*.
Un *graphe* $G = (\mathcal{S}, \mathcal{A})$ est défini par un ensemble $\mathcal{S}$ de *sommets* et un ensemble $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$ d'*arcs*. On dit qu'un arc $(s, t) \in \mathcal{A}$ est de *source* s et de *but* t et qu'il *relie* s à t. Un graphe G est fini si son ensemble de sommets $\mathcal{S}$ est fini.
Le graphe fini $(\{p, q, r, s\}, \{(p, q), (q, r), (r, q)\})$ est représenté ci-dessous. Pour

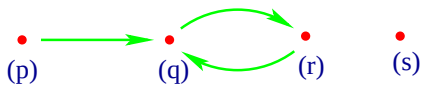


FIG. 2.1 – Un graphe fini.

ce graphe, le sommet s n'est ni source, ni but d'un arc ; on dit alors que s est un *sommet isolé*.
On étend cette notion de graphe en ajoutant de l'information sur les arcs

sous la forme d'*étiquettes*. Ainsi, un graphe est un ensemble d'arcs étiquetés. Soit E un ensemble fini d'étiquettes d'arcs. Un graphe G sur un ensemble de sommets $\mathcal{S}$ est une partie de $\mathcal{S} \times E \times \mathcal{S}$. Tout élément $(p, a, q) \in G$ est un arc de *source* p, de *but* q et d'*étiquette* a. L'ensemble $\mathcal{S}_G$ des *sommets* d'un graphe G est donné par :

$$\mathcal{S}_G \;=\; \{p \in \mathcal{S} \mid \exists a \in E,\ \exists q \in \mathcal{S},\ (p, a, q) \in G \ \vee\ (q, a, p) \in G\}.$$

Dans ce cas, G ne contient pas de sommet isolé. De plus, on considère l'ensemble

$$E_G \;=\; \{a \in E \mid \exists p, q \in \mathcal{S},\ (p, a, q) \in G\}$$

des *étiquettes* du graphe G.
Par exemple le graphe fini $G = \{(p, a, q), (q, b, r), (r, a, q)\}$ est représenté ci-dessous. Un graphe G est *déterministe* si pour chaque sommet s de G tous

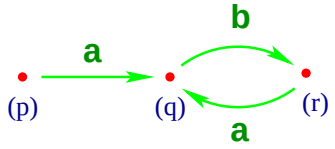


FIG. 2.2 – Un graphe fini étiqueté.

les arcs de source s ont des étiquettes différentes : $\forall s \in \mathcal{S}_G,\ (s, a, p) \in G \wedge (s, a, q) \in G \implies p = q$. Tout arc $(p, a, q) \in G$ pourra être noté $p \xrightarrow[G]{a} q$ ou simplement $p \xrightarrow{a} q$ si le graphe G est sous-entendu. De la même manière que l'on a ajouté de l'information sur les arcs, on peut en ajouter sur les sommets d'un graphe. Soit C un ensemble fini d'étiquettes de sommets. Pour différencier les étiquettes des arcs des étiquettes des sommets, on appellera ces dernières des *couleurs*.
Un graphe G sur un ensemble $\mathcal{S}$ est une partie de $\mathcal{S} \times E \times \mathcal{S} \ \cup\ C \times \mathcal{S}$. Tout élément $(c, p) \in G$, noté aussi cp, est la donnée d'un sommet p colorié par la couleur c.
Prenant a et b des étiquettes d'arcs et i et f des couleurs de sommets, on représente le graphe fini $G = \{(p, a, q), (q, b, r), (r, a, q), ip, fr, is, fs\}$ à la

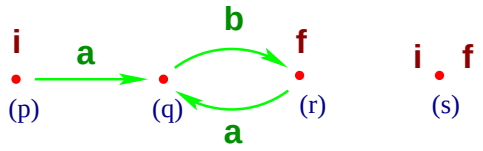


FIG. 2.3 – Un graphe fini étiqueté et colorié.

figure 2.3. Dans un soucis de clarté, on représente un sommet par son nom entre parenthèses afin de le différencier d'une couleur.

Remarquons que dans ce formalisme, on autorise les sommets isolés s'ils sont coloriés ; c'est le cas du sommet s à la figure 2.3.
En plus de l'ensemble $\mathcal{S}_G$ des sommets d'un graphe G et de l'ensemble E_G des étiquettes, on définit l'ensemble

$$C_G \;=\; \{c \mid \exists p \in \mathcal{S}_G,\ cp \in G\}$$

des couleurs des sommets de G, et on note $\mathcal{S}_{G,c}$ l'ensemble des sommets de G coloriés par une couleur c :

$$\mathcal{S}_{G,c} \;:=\; \{s \in \mathcal{S}_G \mid cs \in G\}.$$

Le *degré entrant* (respectivement le *degré sortant*) d'un sommet $s \in \mathcal{S}_G$, $d^-_G(s)$ (resp. $d^+_G(s)$), ou simplement $d^-(s)$ et $d^+(s)$ s'il n'y a pas d'ambiguïté sur le nom du graphe, est le nombre d'arcs dont s est le but (resp. la source) :

$$d^-_G(s) \;:=\; |G \,\cap\, \mathcal{S}_G \times E \times \{s\}| \qquad d^+_G(s) \;:=\; |G \,\cap\, \{s\} \times E \times \mathcal{S}_G|.$$

Le degré $d^\circ(s)$ d'un sommet s est la somme de ses degrés entrant et sortant :

$$d^\circ(s) \;:=\; d^-(s) + d^+(s).$$

Le *degré* $d(G)$ *d'un graphe* G est le maximum des degrés de ses sommets.
Pour le graphe de la figure 2.3, $d^+(p) = 1$, $d^-(q) = 2$ et $d^\circ(s) = 0$.

Un *chemin* d'étiquette $u = a_1 \dots a_n$ dans un graphe G est une suite de n arcs consécutifs de G

$$(p_0, a_1, p_1)(p_1, a_2, p_2) \dots (p_{n-1}, a_n, p_n) \in G^n.$$

Un tel chemin est de *longueur* $n \geq 0$ et est aussi désigné par le mot

$$p_0 a_1 p_1 a_2 p_2 \dots p_{n-1} a_n p_n \in \mathcal{S}_G (E_G \mathcal{S}_G)^*$$

en supposant l'absence d'ambiguïté entre sommets et étiquettes de G *i.e.* $\mathcal{S}_G \cap E_G = \emptyset$.
Si ce chemin existe dans G, on dit que p_n est *accessible* à partir de p_0 (ou p_0 est *co-accessible* à partir de p_n) par un chemin de G étiqueté par u, ce que l'on note $p_0 \overset{u}{\underset{G}{\Longrightarrow}} p_n$. On pourra s'abstraire de l'étiquette $\underset{G}{\Longrightarrow}$ ou du nom du graphe $\overset{u}{\Longrightarrow}$ lorsqu'il n'y a pas d'ambiguïté.
Pour le graphe de la figure 2.3, p est un chemin d'étiquette ε et de longueur 0. De plus, *paqbraq* est un chemin d'étiquette *aba* de longueur 3 allant de p à q.
Un chemin $p_0 a_1 p_1 a_2 p_2 \dots p_{n-1} a_n p_n$ de longueur $n > 0$ ayant les mêmes extrémités $p_0 = p_n$ est appelé un *circuit*.

Un mot $u \in E_G^*$ est une *trace* de G s'il étiquette un chemin entre deux sommets de G. On définit $L(G, I, F)$ l'ensemble des traces de G allant des sommets de I à des sommets de F :

$$L(G, I, F) \ := \ \{u \in E_G^* \mid \exists\ s \in I,\ \exists\ t \in F,\ s \overset{u}{\underset{G}{\Longrightarrow}} t\}.$$

Le *graphe inverse* G^{-1} de G est l'ensemble des sommets coloriés de G et des arcs inverses de G :

$$G^{-1} \ := \ \{t \xrightarrow{a} s \mid s \xrightarrow{a} t\} \ \cup \ \{cs \in G \mid c \in \mathcal{F}_1 \wedge s \in \mathcal{S}_G\}.$$

Une *chaîne* (respectivement un *cycle*) de G est un chemin (respectivement un circuit) de $G \cup G^{-1}$. La *distance* $d_G(s, t)$ entre deux sommets $s, t \in \mathcal{S}_G$ d'un graphe G est la plus petite longueur des chaînes de G entre s et t :

$$d_G(s, t) \ := \ min(\{|u| \mid s \overset{u}{\underset{G \cup G^{-1}}{\Longrightarrow}} t\} \ \cup \ \{\omega\}).$$

Si tout sommet $p \in \mathcal{S}_G$ est accessible à partir d'un sommet r :

$$r \underset{G}{\Longrightarrow} p \qquad \text{pour tout } p \in \mathcal{S}_G,$$

on dit de r qu'il est une *racine* de G.
La *restriction* $G_{|P}$ d'un graphe G à un ensemble P de sommets est le sous-graphe induit par P :

$$G_{|P} \;=\; \{(s,a,t) \in G \mid s,t \in P\} \cup \{cp \in G \mid p \in P\}.$$

Le *sous-graphe accessible* à partir d'un sommet s est la restriction de G aux sommets accessibles à partir de s :

$$G_{\downarrow s} \;=\; G_{|\{q \mid s \Longrightarrow q\}}.$$

On définit également le sous-graphe de G accessible (resp. co-accessible) à partir d'une couleur c. $G_{c\rightarrow}$ (resp. $G_{\rightarrow c}$) est la restriction de G aux sommets accessibles (resp. co-accessibles) à partir des sommets coloriés par c :

$$\begin{aligned} G_{c\rightarrow} &= \bigcup_{cs \in G} G_{\downarrow s} \\ G_{\rightarrow c} &= G_{|\{s \in G \mid \exists t,\ s \underset{G}{\Longrightarrow} t \wedge ct \in G\}} \end{aligned}$$

et on introduit la notation $G_{c_1 \rightarrow c_2}$ pour désigner la restriction de G aux sommets accessibles par la couleur c_1 et co-accessibles par la couleur c_2.

Chapitre 3

Des automates finis

Notre étude concerne les langages de mots. Il existe différents types de langages que l'on regroupe par classes. On peut citer notamment les langages réguliers ou encore les langages algébriques. Pour chaque classe de langages, il existe une classe de générateur : les grammaires de mots. Dans le premier paragraphe, après avoir rappelé ce que sont les grammaires de mots, nous décrirons la classification des langages selon Chomsky [Ch56]. Les grammaires de mots ne sont pas les seuls objets qui nous permettent de travailler sur les langages. Nous utilisons également des graphes coloriés et étiquetés : les automates. Pour chaque classe de langage, on décrit une classe d'automates de sorte que les traces de ces automates correspondent aux langages. Dans le paragraphe 2 de ce chapitre, nous décrivons les automates finis qui reconnaissent les langages réguliers, l'étude des automates associés aux langages algébriques se faisant au chapitre suivant.

3.1 Grammaires de mots

Dans ce paragraphe, nous décrivons un objet qui permet d'engendrer les langages de mots : les grammaires de mots. Ces langages (et les grammaires associés) peuvent être classifiées en fonction de leur compléxité. Ceci a été fait par Noam Chomsky en 1956.

3.1.1 Définition

Une *grammaire* $R = (\Sigma, \mathcal{N}, D, \delta)$ est un quadruplet où

- Σ est un alphabet de symboles dits *terminaux* (notés par des lettres minuscules),
- $\mathcal{N}$ est un alphabet de symboles dits *non-terminaux* (notés par des lettres majuscules),
- un non-terminal de départ D appelé *axiome*,
- un ensemble fini δ de *règles* de la forme

$$u \rightarrow w$$

telles que $u \in (\mathcal{N} \cup \Sigma)^+$ et $w \in (\mathcal{N} \cup \Sigma)^*$;

et on appelle u le *membre gauche* de la règle et w son *membre droit*.
Ainsi, δ est une relation binaire sur les mots :

$$\delta \subseteq (\mathcal{N} \cup \Sigma)^+ \times (\mathcal{N} \cup \Sigma)^*.$$

Dans l'exemple suivant, nous donnons une grammaire et une représentation schématique des règles.

Exemple 3.1.1. La grammaire $R = (\{a, b\}, \{A\}, A, \{(A, \varepsilon), (A, aAb)\})$ est un premier exemple de grammaire. Pour plus de lisibilité, nous décrivons ses règles à la figure 3.1.

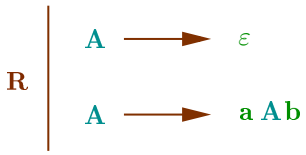


FIG. 3.1 – Une grammaire de mots.

On différencie les grammaires de mots en fonction de la forme de leurs règles. Ceci a été fait par Noam Chomsky [Ch56] et est discuté au paragraphe suivant.

L'ensemble δ des règles définit une relation binaire de *récriture* sur $(\Sigma \cup \mathcal{N})^*$. Un mot $u = u'xu''$ se *récrit* selon la grammaire R en un mot $v = u'wu''$, et on note

$$u = u'xu'' \underset{R}{\longrightarrow} u'wu'' = v$$

si $x \longrightarrow w$ est une règle de R, *i.e.* $(x, w) \in \delta$.
Dans l'exemple suivant, nous détaillons les premières récritures en fonction des règles de l'exemple 3.1.

Exemple 3.1.2. Les récritures suivant la règle (A, aAb) sont représentées horizontalement et celles suivant (A, ε) le sont verticalement.

$$\begin{array}{ccccccccc} \mathrm{A} & \rightarrow & \mathrm{aAb} & \rightarrow & \mathrm{aaAbb} & \rightarrow & \mathrm{aaaAbbb} & & \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & \cdots & \\ \varepsilon & & \mathrm{ab} & & \mathrm{aabb} & & \mathrm{aaabbb} & & \end{array}$$

La fermeture réflexive et transitive de la relation de récriture (suivant une grammaire R) est notée $\underset{R}{\longrightarrow}^*$ et on parle alors de *dérivation* (selon R). Le mot u se dérive en le mot v, noté $u \underset{R}{\longrightarrow}^* v$, si il existe une suite de $n \geq 0$ récritures selon R de u à v :

$$u = u_0 \underset{R}{\longrightarrow} \ldots \underset{R}{\longrightarrow} u_n = v.$$

Le *langage engendré* $L(R)$ par une grammaire $R = (\Sigma, \mathcal{N}, D, \delta)$ est l'ensemble des mots de Σ^* qui peuvent se dériver à partir de l'axiome D :

$$L(R) \;=\; \{u \in \Sigma^* \mid D \underset{R}{\longrightarrow}^* u\}$$

Le langage engendré par la grammaire de l'exemple 3.1 est $\{a^n b^n \mid n \geq 0\}$; autrement dit, cette grammaire engendre tous les mots commençant par n occurrences de la lettre a, suivies par n occurrences de la lettre b.

3.1.2 La hiérarchie de Chomsky

En 1956, Noam Chomsky [Ch56] classa les langages en fonction de leur complexité d'expression. Il définit quatre classes de langages de compléxité

croissante, constituant ainsi la *hiérarchie de Chomsky*.
A chacune de ces classes, on fait correspondre un type de grammaire de mots engendrant la famille de langages. Ces types de grammaires se différencient par la structure syntaxique des règles. Ces informations sont regroupées dans le tableau suivant,

Niveau	Langage	Grammaire	Forme des règles
3	régulier	linéaire droite ou gauche	$A \to aB$ ou $A \to Ba$
2	algébrique	algébrique	$A \to w$
1	contextuel	contextuelle	$sAt \to sut$
0	récursivement énumérable	générale	$v \to w$

dans lequel $A, B \in \mathcal{N}$, $s, t \in \Sigma^*$, $u, v, w \in (\Sigma \cup \mathcal{N})^*$ et $u \neq \varepsilon$.

On peut distinguer des sous-classes dans cette hiérarchie. Une subdivision est donnée à la figure 3.2, avec les notations suivantes :

- RE est la classe des langages récursivement énumérables (acceptés par les machines de Turing),
- $CONT$ est la classe des langages contextuels (acceptés par les machines de Turing linéairement bornés),
- ALG est la classe des langages algébriques (acceptés par les automates à pile),
- $DALG$ est la classe des langages algébriques déterministes (acceptés par les automates à pile déterministes),
- REG est la classe des langages réguliers (acceptés par les automates finis) et
- FIN est la classe des langages finis.

Dans ce document, nous nous intéressons principalement aux niveaux trois et deux de la hiérarchie, c'est-à-dire aux langages réguliers et aux langages algébriques. Les langages étant généralement des ensembles infinis, il n'est pas possible de les spécifier en donnant tous leurs mots. C'est pourquoi nous utilisons des outils finis pour les décrire, afin ainsi de dégager des propriétés

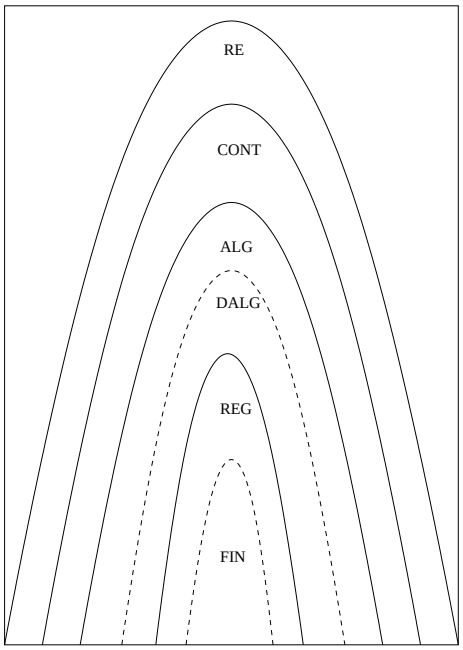


FIG. 3.2 – La hiérarchie de Chomsky

sur les langages en observant les propriétés de leur description.
Nous verrons dans le prochain paragraphe l'utilité des automates finis pour travailler sur les langages réguliers. Cette classe des langages possède bien des propriétés de décidabilité comme par exemple la clôture par intersection et par complément, ce qui permet notamment de décider de l'inclusion (la vacuité d'un langage régulier étant décidable). Mais ces propriétés ne sont pas nécessairement valables pour toutes les autres classes de la hiérarchie de Chomsky. Par exemple la famille des langages contextuels possède les propriétés de clôture des langages réguliers mais la vacuité de ces langages n'est pas décidable. Un autre exemple est la classe des langages algébriques qui n'est pas close par intersection. Néanmoins, nous montrons qu'il est possible de généraliser les propriétés de clôture et de décidabilité des langages réguliers à des sous-classes (strictes) de langages algébriques en utilisant une méthode dite de synchronisation. Après avoir décrit les propriétés qui nous intéressent sur les langages réguliers, nous verrons comment les étendre.

3.2 Automates finis

Dans la hiérarchie de Chomsky, les langages réguliers sont au niveau 3. Ils sont parmi les langages infinis ceux qui ont le plus petit pouvoir d'expression (*cf.* exemple 3.2.1).
Pour un alphabet Σ, on note $Reg(\Sigma^*)$ l'ensemble des langages réguliers dont les mots sont construits sur l'alphabet Σ.
Rappelons que $Reg(\Sigma^*)$ est la plus petite famille de langages telle que, pour tout langage $L, L' \in Reg(\Sigma^*)$:

- $\emptyset$ est un langage régulier ;
- $\forall a \in \Sigma,\ \{a\} \in Reg(\Sigma^*)$;
- $Reg(\Sigma^*)$ est clos par *union* : $L \cup L' = \{u \mid u \in L \vee u \in L'\} \in Reg(\Sigma^*)$;
- $Reg(\Sigma^*)$ est clos par *concaténation* :
$$L.L' = \{uv \mid u \in L \ \wedge \ v \in L'\} \in Reg(\Sigma^*),$$
- $Reg(\Sigma^*)$ est clos par l'*étoile de Kleene* qui est la fermeture réflexive et transitive de la concaténation :
$$L^* = \{u_1 \dots u_n \mid n \geq 0 \ \wedge \ u_1, \dots, u_n \in L\} \in Reg(\Sigma^*).$$

Exemple 3.2.1. Le langage Σ^* de tous les mots sur Σ est un langage régulier. Sur l'alphabet $\{a, b\}$, le langage $L = a^*b^* = \{a^m b^n \mid m, n \geq 0\}$ des mots commençant par une suite de a de longueur quelconque, suivie par une suite de b de longueur quelconque est aussi un langage régulier.
Mais le langage $L' = \{a^n b^n \mid n \geq 0\} \subset L$ n'est pas régulier : on ne peut pas exprimer l'égalité par un langage régulier.

Il existe bien des façons de définir un langage régulier. Nous donnons ici deux types de machines abstraites pour reconnaître les langages réguliers, à savoir les grammaires linéaires gauches et les automates finis.

3.2.1 Les grammaires linéaires gauches

En théorie des langages, il est bien connu que les langages réguliers sont les langages engendrés par les grammaires linéaires gauches.
Une grammaire de mots $R = (\Sigma, \mathcal{N}, D, \delta)$ est *linéaire gauche* si chaque

membre gauche de ses règles est un unique non-terminal et chaque membre droit est un mot terminal ou un mot composé d'un non-terminal suivi par un mot terminal :

$$\forall (A, w) \in \delta,\ A \in \mathcal{N}\ \wedge\ w \in \mathcal{N}.\Sigma^*\ \cup\ \Sigma^*$$

où . est la concaténation sur $(\mathcal{N} \cup \Sigma)^*$.

Remarque : Dans la littérature, il existe également les grammaires linéaires droites pour lesquelles, dans le membre droit des règles, le non-terminal se trouve à droite du mot terminal. Comme les deux modèles sont équivalents, nous en avons choisi un arbitrairement.

Les grammaires de mots offrent une façon de spécifier la structure syntaxique des langages. Une autre façon de décrire ou reconnaître les langages réguliers est l'utilisation des traces des automates finis. Ceux-ci permettent en effet de donner une manière géométrique de description des langages réguliers.

3.2.2 Les automates finis

Un *automate fini* est une machine abstraite composée d'*états* et de *transitions*. Un automate fini $M = (\mathcal{S}, \mathcal{A}, \Sigma, I, F)$ est la donnée

- d'un ensemble fini $\mathcal{S}$ d'états,
- d'un ensemble $\mathcal{A}$ de transitions entre les états, étiquetées sur Σ : $\mathcal{A} \subseteq \mathcal{S} \times \Sigma \times \mathcal{S}$,
- d'un ensemble $I \subseteq \mathcal{S}$ d'états initiaux,
- et d'un ensemble $F \subseteq \mathcal{S}$ d'états finaux.

Tout automate fini $M = (\mathcal{S}, \mathcal{A}, \Sigma, I, F)$ est le graphe colorié et étiqueté $G = (\mathcal{S}, \mathcal{A})$ tel que

- les sommets correspondants aux états initiaux sont coloriés par une couleur particulière, disons $i : I = \mathcal{S}_{G,i}$ et
- les sommets correspondants aux états finaux sont coloriés par une couleur particulière, disons $f : F = \mathcal{S}_{G,f}$.

Un automate $M = (\mathcal{S}, \mathcal{A}, \Sigma, I, F)$ est *déterministe* si son graphe $(\mathcal{S}, \mathcal{A})$ est déterministe et qu'il ne possède qu'un seul état initial $|I| = 1$. Par la suite,

nous identifierons un automate fini avec son graphe colorié.
Nous utilisons ici les automates comme accepteurs de langages. Un langage $L(M)$ construit sur Σ est *accepté* (ou *reconnu*) par un automate $M = (\mathcal{S}, \mathcal{A})$ s'il est l'ensemble des traces de M entre les sommets initiaux et ceux finaux :

$$L(M) \;=\; L(M, I, F) \;\subseteq\; \Sigma^*.$$

Les langages réguliers qui ont été définis par clôture des langages finis par union, concaténation et étoile, sont exactement les langages acceptés par les automates finis. Ce résultat est dû à Kleene [Kl56].

Théorème 3.2.2. *Les langages réguliers sont les langages acceptés par les automates finis.*

On peut transformer tout automate fini en un automate fini déterministe et complet équivalent.

Propriété 3.2.3. *Tout langage régulier est accepté par un automate fini déterministe et complet.*

Par l'opération de déterminisation Det appliquée à un graphe fini G, on construit le graphe fini $Det(G)$ déterministe tel que $L(G) = L(Det(G))$:

$$\begin{array}{rcl} Det(G) & := & \{S_1 \xrightarrow{a} S_2 \mid S_1 \subseteq \mathcal{S}_G \wedge\ \forall t\ (t \in S_2 \iff \exists s \in S_1,\ s \xrightarrow[G]{a} t)\} \\ & \cup & \{i\{p \mid ip \in G\}\} \\ & \cup & \{fP \mid P \subseteq \mathcal{S}_G \wedge \exists p \in P,\ fp \in G\} \end{array}$$

La propriété est effective : tout automate fini M est transformé en un automate fini M' déterministe et complet tel que $L(M) \;=\; L(M')$. La construction est illustrée à la figure 3.3.
Notons que, de manière générale, nous ne représentons pas les sommets non accessibles à partir du sommet initial. C'est en utilisant les automates finis que nous allons décrire les propriétés de clôture par intersection et complémentation des langages réguliers. Nous les étendrons ensuite à des sous-familles de langages algébriques.

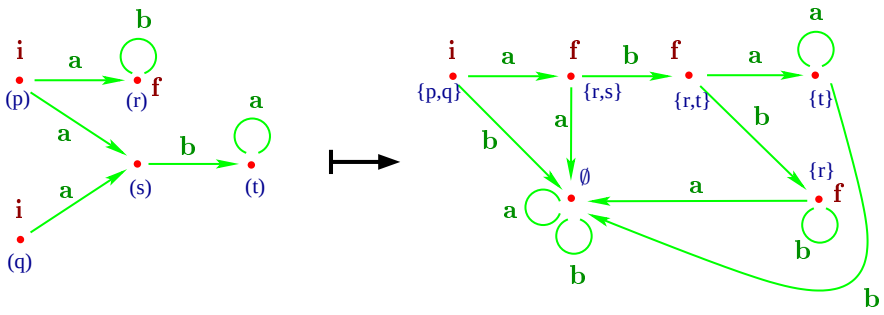


FIG. 3.3 – Déterminisation d'un automate fini.

3.2.3 Propriétés de fermeture

Par définition, la famille des langages réguliers sur un alphabet Σ est close par union, par concaténation et par l'étoile de Kleene. Elle est aussi fermée par intersection et par complémentaire.
Nous donnons ici les constructions habituelles sur les automates finis pour la clôture par intersection, par complémentaire, mais aussi pour la clôture par concaténation et sa fermeture réflexive et transitive.

L'intersection

Soient L_1 et L_2 les langages réguliers sur Σ reconnus respectivement par des automates finis G et H. Nous construisons un automate fini qui accepte le langage $L_1 \cap L_2$. Pour cela on utilise le *produit de synchronisation* $G \times H$ de G par H défini comme suit :

$$\begin{aligned} G \times H \;&:=\; \{(s,p) \xrightarrow{a} (t,q) \mid s \xrightarrow[G]{a} t \;\wedge\; p \xrightarrow[H]{a} q\} \\ &\cup\; \{i(s,p) \mid is \in G \;\wedge\; ip \in H\} \\ &\cup\; \{f(s,p) \mid fs \in G \;\wedge\; fp \in H\} \end{aligned}$$

L'automate ainsi construit est fini et on a

$$L(G \times H, i, f) \;=\; L_1 \cap L_2.$$

Ainsi, $Reg(\Sigma^*)$ est clos par intersection.
Prenons les automates G et H représentés à la figure 3.4 qui reconnaissent les langages $L(G) = a(ba)^*b$ et $L(H) = a^*b(a+b)^*$. On a $L(G) \subseteq L(H)$.

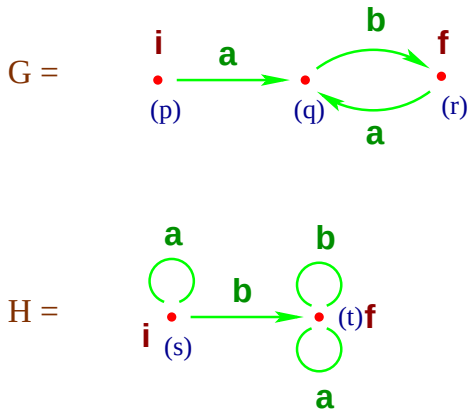


FIG. 3.4 – Deux automates finis G et H.

La figure 3.5 illustre la construction de l'automate $G \times H$ qui reconnaît le langage $L(G) \cap L(H) = L(G)$ mais n'est pas isomorphe à G.

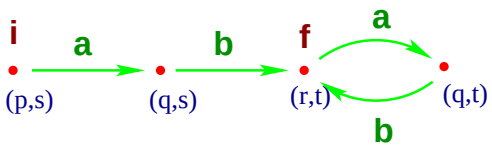


FIG. 3.5 – L'automate $G \times H$.

Le complémentaire

Soit G un automate fini reconnaissant le langage régulier $L = L(G)$. Nous voulons construire un automate reconnaissant le complémentaire de L, à savoir $\Sigma^* - L = \{u \in \Sigma^* \mid u \notin L\}$.

La construction se fait en deux étapes :

- on construit l'automate H déterministe et complet qui reconnaît le langage $L(G)$,
- on construit l'automate $\overline{H}$ en interchangeant les sommets finaux et non-finaux de H :

$$\overline{H} = (H - f\mathcal{S}_H) \cup \{fs \mid s \in \mathcal{S}_H \wedge fs \notin H\}.$$

L'automate $\overline{H}$ est fini et reconnaît le complémentaire de L.

Pour l'automate G défini à la figure 3.3, l'automate $\overline{H}$ de la figure 3.6 reconnaît le complémentaire de $L(A)$.

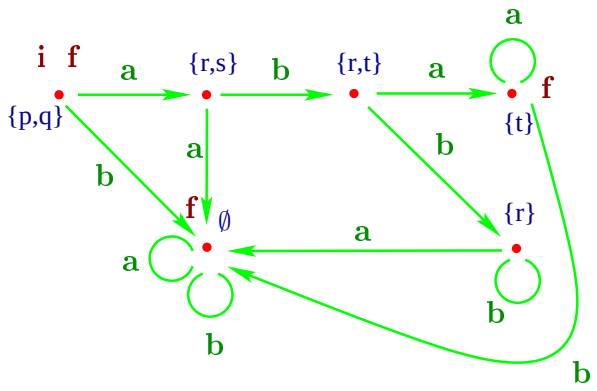


FIG. 3.6 – L'automate complément de celui à la figure 3.3.

La concaténation

Soient L_1 et L_2 deux langages réguliers reconnus respectivement par des automates finis G et H. On peut supposer que les sommets de G et de H sont distincts : $\mathcal{S}_G \cap \mathcal{S}_H = \emptyset$.
Pour construire l'automate qui accepte $L_1.L_2$, on relie les sommets finaux de G aux buts des transitions initiales de H (en leur retirant leurs couleurs si $\varepsilon \notin L_2$) :

$$\begin{aligned} G.H \;&:=\; (G - \{fs \in G \mid \varepsilon \notin L(H)\}) \\ &\cup\; (H - i\mathcal{S}_H) \\ &\cup\; \{s \xrightarrow{a} t \mid fs \in G \wedge \exists p,\ ip \in H \wedge p \xrightarrow[H]{a} t\} \end{aligned}$$

En prenant les automates G et H définis à la figure 3.4, on construit l'automate $G.H$ de la figure 3.7.

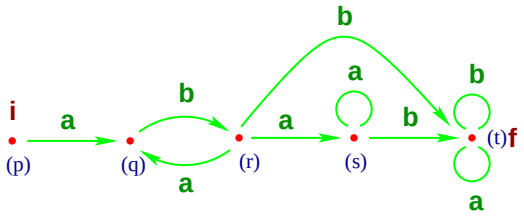


FIG. 3.7 – L'automate $G.H$.

On peut remarquer que l'automate qui reconnaît la concaténation des langages peut ne pas être déterministe, et cela même si les automates reconnaissant ces langages le sont. Mais on peut toujours le déterminiser.
En appliquant *Det* à l'automate de l'exemple précédent, on obtient l'automate déterministe de la figure 3.8 où on s'est restreint aux sommets coaccessibles à partir des sommets finaux.

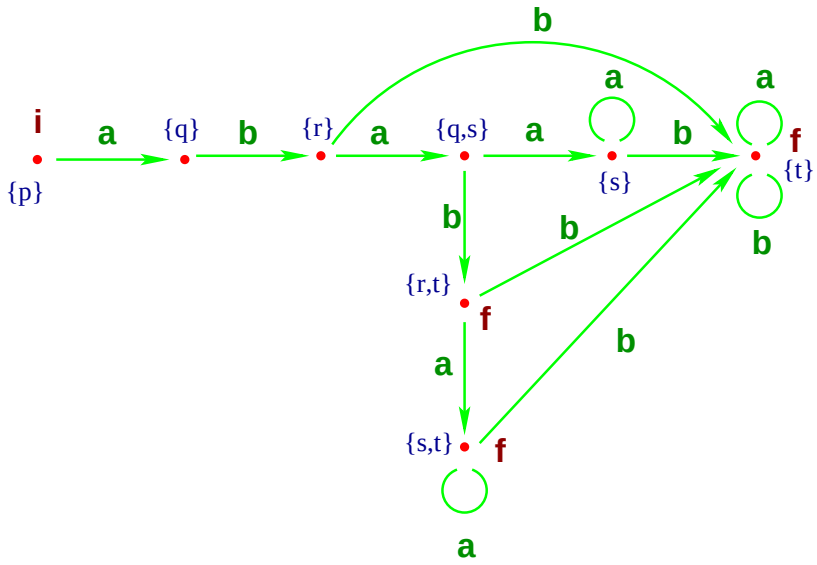


FIG. 3.8 – Déterminisation de l'automate de la figure 3.7.

L'étoile de Kleene

Soit L un langage régulier reconnu par un automate fini G (que l'on peut supposer déterministe). Nous cherchons à construire un automate fini G^* acceptant le langage L^*. Pour cela, on transforme l'automate G en un automate équivalent G' de sorte que le sommet initial de G' ne soit but d'aucun arc. Soit p un nouveau sommet :

$$\begin{aligned} G' &:= (G - \{is \mid is \in G\}) \cup \{ip\} \\ &\cup \{p \xrightarrow{a} t \mid \exists s\ (is \in G \ \wedge\ s \xrightarrow{a} t \in G)\} \end{aligned}$$

A partir de G', on construit G^+ : pour tout arc ayant pour but un sommet final on ajoute un arc de même source et ayant pour but le sommet initial de G' :

$$G^+ \quad := \quad G' \cup \{s \xrightarrow{a} p \mid \exists t,\ ft \in G' \wedge s \xrightarrow{a} t\}$$

Cette grammaire reconnaît le langage L^+. Il ne reste plus qu'à rajouter l'acceptation du mot vide :

$$G^* \quad := \quad G^+ \cup \{fp\}.$$

G^* est donc bien un automate fini reconnaissant $L(G)^*$. A titre d'exemple, considérons l'automate fini H de la figure 3.4. L'automate obtenu par cette construction et donc acceptant $L(H)^*$ est représenté à la figure 3.9.

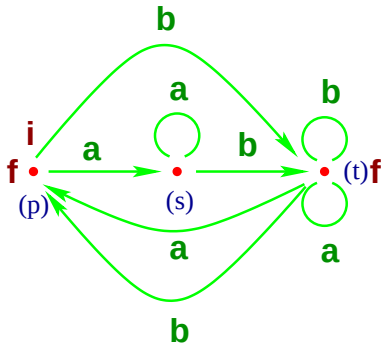


FIG. 3.9 – L'automate H^*.

L'utilisation des automates finis dans l'étude des langages réguliers nous donne une méthode efficace et simple d'obtenir l'effectivité des propriétés de clôture de ces langages. Dans le chapitre suivant, nous décrivons les langages algébriques. Cette classe de langages n'étant pas close pour plusieurs des opérations présentées ici, ces constructions ne leur sont pas extensibles. Néanmoins, nous verrons dans le chapitre 4 comment définir des sous-classes strictes de langages algébriques pour lesquelles on étendra les constructions de ce chapitre pour prouver leurs propriétés de clôture par les opérations que nous avons décrites.

Chapitre 4

Aux automates réguliers

4.1 Automates à piles

Dans le chapitre précédent, nous avons traité des langages acceptés par les automates à états finis : les langages réguliers. Dans la hiérarchie de Chomsky, ils sont strictement inclus dans les *langages algébriques*. De même que pour les langages réguliers, il existe différents formalismes finis pour engendrer (ou reconnaître) les langages algébriques. Nous donnons la manière historique de les construire à savoir à l'aide des grammaires algébriques (de mots). Nous voyons ensuite que ces grammaires de mots ont le même pouvoir d'expression que les automates à pile. En représentant les transitions de ces machines par des graphes, on définit la classe des graphes réguliers. Une propriété géométrique de ces graphes a permis de définir un type de grammaires sur les graphes, qui étend les grammaires de mots.

4.1.1 Les grammaires algébriques

Un langage est algébrique s'il est engendré par une grammaire de mots dite *algébrique*. Les règles de ces grammaires sont telles que tout membre gauche est réduit à une lettre dite non-terminale et que tout membre droit est un mot composé de non-terminaux et d'autres lettres dites terminales. La grammaire $R = (\Sigma, \mathcal{N}, D, \delta)$ telle que

$$\forall (A, u) \in \delta,\ A \in \mathcal{N} \ \wedge \ u \in (\mathcal{N} \cup \Sigma)^*$$

est algébrique. La grammaire représentée à la figure 3.1 est algébrique et engendre le langage algébrique $L(R) = \{a^n b^n \in \Sigma^* \mid n \geq 0\}$.

4.1.2 Les automates à piles

Un *automate à pile* est une machine abstraite composée d'une bande de travail munie d'une tête de lecture à états finis et d'une mémoire sous la forme d'une pile LIFO (Last In First Out *i.e.* dernier entré, premier sorti). Les mouvements de la tête de lecture sont uni-directionnels et sont conditionnés par un ensemble de règles appelé relation de transition (ou également *programme* de la machine). La figure 4.1 représente un automate à pile dont la tête de lecture dans l'état p lit une lettre a sur la bande de travail et possède le mot AU dans sa pile, avec A en sommet de pile.

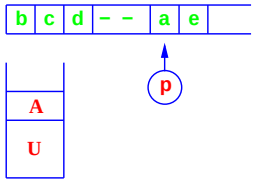


FIG. 4.1 – Un automate à pile.

Nous réservons un *symbole de fond de pile*, $\perp$, que l'on ne peut ni empiler ni dépiler.
Un automate à pile $M = (S, \Gamma, \Sigma, s_0, F, \delta)$ est défini par

- un ensemble fini S d'*états*,
- un alphabet Γ de lettres de pile tel que $\perp \notin \Gamma$,
- un alphabet Σ d'étiquettes,
- un *état initial* $s_0 \in S$,
- un ensemble $F \subseteq S$ d'*états finaux*,
- une relation de *transition* δ entre les *configurations* de l'automate.

Une *configuration* d'un automate à pile est un mot de $S\Gamma^*\perp$ décrivant l'état de l'automate et le mot de sa pile.

Ayant $p, q \in S$, $a \in \Sigma \cup \{\varepsilon\}$, $A \in \Gamma \cup \{\perp\}$ et $B \in \Gamma$, la relation de transition δ est donnée comme un ensemble fini de règles normalisées de la

forme

$$pA \xrightarrow{a} qBA$$

où l'on empile la lettre B si le sommet de pile est la lettre A, ou

$$pA \xrightarrow{a} q \qquad \text{avec } A \neq \bot$$

où l'on dépile la lettre $A \neq \bot$ en sommet de pile, ou encore

$$pA \xrightarrow{a} qA$$

où l'on effectue un changement d'état sans modifier la pile.
Les règles de la première forme sont qualifiées de règles d'*ajout*, celles de la seconde forme de règles de *retrait* (mais $\bot$ ne peut être dépilée) et celles de la dernière forme de règles d'*action interne*.

Exemple 4.1.1. Soit l'automate à pile M suivant :

$$M = (\{q_0,\ q_1,\ q_2,\ q_f\}, \{A\}, \{a, b, \varepsilon\}, q_0, \{q_f\}, \delta_M)$$

avec δ_M donné par les règles suivantes :

$$\begin{aligned} q_0\bot &\xrightarrow{a} q_1A\bot \\ q_1A &\xrightarrow{a} q_1AA \\ q_1A &\xrightarrow{b} q_2 \\ q_2A &\xrightarrow{b} q_2 \\ q_2\bot &\xrightarrow{\varepsilon} q_f\bot \end{aligned}$$

Pour la règle d'ajout $pA \xrightarrow{a} qBA$, δ réalise la transition suivante :

$$pAU \xrightarrow{a} qBAU \quad (ajout)$$

où $AU \in \Gamma^*\bot$. De même l'application d'une règle de retrait $pA \xrightarrow{a} q$ fait passer l'automate de la configuration pAU à la configuration qU, et une action interne $pA \xrightarrow{a} qA$ déclenchera le passage de la configuration pAU à la configuration qAU :

$$\begin{aligned} pAU &\xrightarrow{a} qU \quad (retrait) \\ pU &\xrightarrow{a} qU \quad (action\ interne) \end{aligned}$$

Prenant $M = (S, \Gamma, \Sigma, s_0, F, \delta)$ un automate à pile, nous définissons son *graphe des transitions* dont les sommets sont des configurations de M :

$$H_M \;=\; \{pAU \xrightarrow{a} qVU \mid pA \xrightarrow{a} qV \in \delta \;\wedge\; AU \in \Gamma^*\bot\}.$$

Plus particulièrement, nous nous intéressons aux configurations accessibles par la relation de transition à partir de la configuration initiale $s_0\bot$ et nous notons

$$G_M \;:=\; (H_M \cup \{i(s_0\bot)\} \cup \{f(sU) \mid s \in F \wedge U \in \Gamma^*\bot\})_{\downarrow(s_0\bot)}$$

le *graphe de transition* des configurations de l'automate M.
Notons que cette définition d'automate à pile correspond dans la littérature aux automates à piles dits »faibles « : chaque transition ne modifie la taille de la pile d'au plus un. Mais les automates à pile faibles et les automates à pile définissent les mêmes graphes de transition (à isomorphisme près) [Ca06].
Le langage $L(M)$ *reconnu* par M est l'ensemble des traces entre le sommet colorié par i et les sommets coloriés par f :

$$L(M) \;=\; L(G_M, i, f).$$

Un automate à pile M est *complet* si $L(M) = \Sigma^*$.
Le graphe de transition de l'automate à pile M défini à l'exemple 4.1.1 est représenté à la figure 4.2, et $L(M) \;=\; \{a^n b^n \mid n > 0\}$.

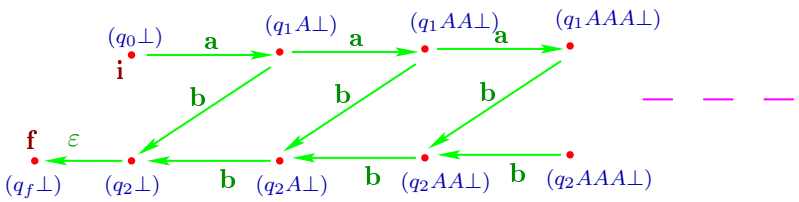


FIG. 4.2 – Graphe de transition de l'automate à pile de l'exemple 4.1.1.

Les automates à pile reconnaissent les langages engendrés par les grammaires algébriques.

Propriété 4.1.2. *Les langages reconnus par les automates à piles sont exactement les langages algébriques.*

Un automate à pile est *déterministe* si

- d'une part pour chaque configuration, la configuration que l'on obtient par la relation de transition est unique :
$$pA \xrightarrow{a} qV \in \delta \ \wedge \ pA \xrightarrow{a} q'V' \in \delta \implies q = q' \ \wedge \ V = V'$$
- et d'autre part à partir d'une configuration, on ne peut avoir que soit des transitions étiquetées par des lettres de $\Sigma - \{\varepsilon\}$, soit des transitions étiquetées par ε :
$$pA \xrightarrow{a} qV \in \delta \ \wedge \ pA \xrightarrow{\varepsilon} q'V' \in \delta \implies a = \varepsilon.$$

Les langages engendrés par les automates à pile déterministes sont les *langages algébriques déterministes*.

Un automate à pile est *temps réel* s'il ne possède aucune transition étiquetée par ε :

$$c \xrightarrow{a} c' \in \delta \implies a \in \Sigma - \{\varepsilon\}.$$

Les automates à pile temps réel reconnaissent les langages algébriques.

Ces diverses restrictions d'automates à pile reconnaissent des sous-familles différentes de langages algébriques, comme indiqué par le schéma ci-dessous :

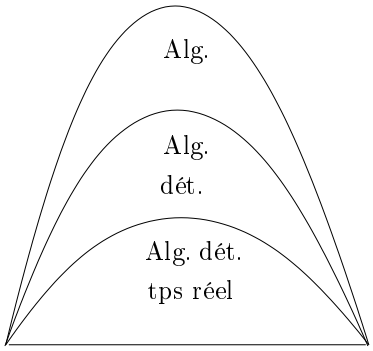


FIG. 4.3 – Sous-familles de langages algébriques.

Le langage $\{a^n b^n \mid n \geq 0\}$ accepté par l'automate de l'exemple 4.1.1 est un langage algébrique déterministe et temps réel.

Le langage $\{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0\}$ est un langage algébrique non déterministe.

Donnons un langage algébrique déterministe mais non déterministe temps réel. Considérons l'automate à pile déterministe M suivant :

$$
\begin{array}{llll|llll|llll}
i\bot & \xrightarrow{a} & pA\bot & & pA & \xrightarrow{a} & pAA & & pB & \xrightarrow{c} & pBB \\
qA & \xrightarrow{\varepsilon} & pA & & pA & \xrightarrow{b} & p & & pB & \xrightarrow{d} & p \\
qB & \xrightarrow{\varepsilon} & q & & pA & \xrightarrow{c} & pBA & & pB & \xrightarrow{e} & qB \\
p\bot & \xrightarrow{\varepsilon} & i\bot & & pA & \xrightarrow{e} & qA & & & &
\end{array}
$$

d'état initial et final i. Son graphe de transition G_M est représenté comme suit :

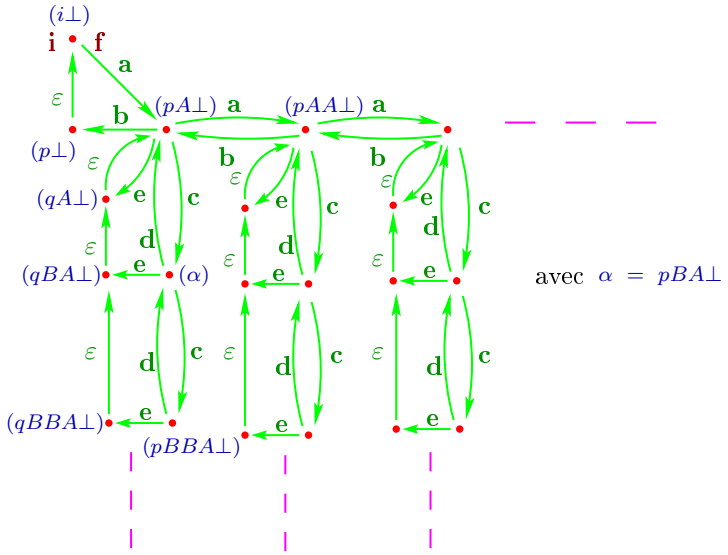


FIG. 4.4 – G_M est déterministe et non temps réel.

Le langage $L(M)$ reconnu par M est un langage algébrique déterministe mais il ne peut pas être reconnu par un automate à pile déterministe temps réel.

Indiquons que la forme particulière des règles des automates à piles (ajout, retrait, action interne) n'est pas une restriction pour les langages engendrés, même en se restreignant aux automates déterministes et/ou temps réel. En fait, ils définissent les mêmes familles de graphes de transition [Ca06].

Muller et Schupp furent les premiers à étudier les automates à pile du point de vue de la géométrie de leurs graphes de transition [MS85]. A leur

sens, un graphe de degré fini est régulier si on peut le décomposer par distance à partir d'un sommet en un nombre fini de composantes connexes (à isomorphisme près).
Pour une meilleure compréhension des graphes que nous utilisons dans ce document, attardons nous sur cette notion de décomposition finie par distance. On se restreint aux graphes G dont tout sommet est connecté (par une chaîne) à un sommet initial : toute composante connexe de G possède au moins un sommet initial. Dans ce cas, la distance de tout sommet s de G définie par

$$\begin{aligned} d_G(s) &:= min\{d_G(s,t) \mid it \in G\} \\ &= min\{|u| \mid \exists t\ (it \in G \wedge s \underset{G \cup G^{-1}}{\overset{u}{\Longrightarrow}} t)\} \end{aligned}$$

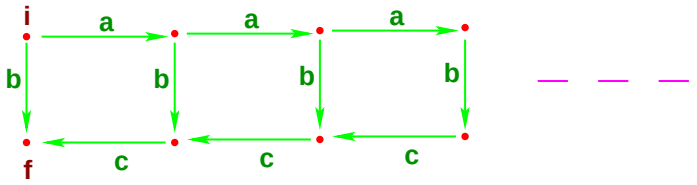
est finie. En particulier, les sommets de distance nulle sont les sommets initiaux : $d_G(s) = 0 \iff is \in G$.
Définissons la décomposition de G à partir de ses sommets initiaux. La première étape de la décomposition de G consiste à retirer les sommets initiaux de G, à savoir les couleurs et les arcs portant sur un sommet initial, et à prendre pour sommet initial tout nouveau sommet d'un arc retiré ; autrement dit, on obtient le graphe suivant :

$$\widehat{G} := G_{|\{s \in \mathcal{S}_G \mid is \notin G\}} \cup \{is \mid is \notin G \wedge \exists t\ (s \underset{G \cup G^{-1}}{\longrightarrow} t \wedge it \in G)\}.$$

Puis, itérativement, on continue la décomposition à partir du graphe $\widehat{G}$ obtenu, à savoir :

$$\widehat{G}^0 := G \qquad \text{et} \qquad \widehat{G}^{n+1} = \widehat{\widehat{G}^n} \text{ pour tout } n \geq 0.$$

On dit que G est de *décomposition finie par distance* si $\{\widehat{G}^n \mid n \geq 0\}$ n'a qu'un nombre fini de composantes connexes non isomorphes.
Par exemple, le graphe suivant :



est de décomposition finie par distance n'ayant par décomposition que les trois composantes connexes ci-après :

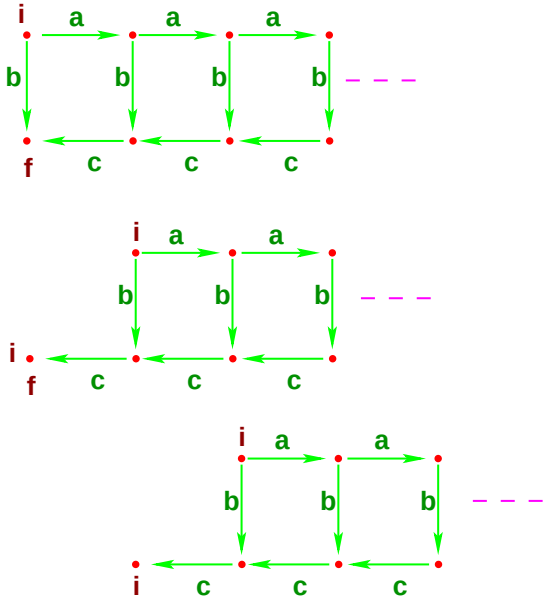
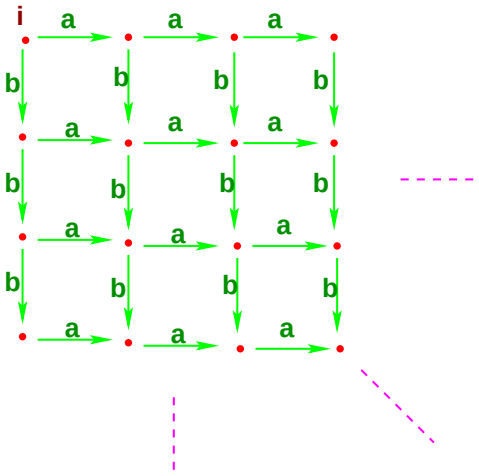


FIG. 4.5 – Décomposition par distance d'un graphe.

Par contre, la grille ci-après n'admet pas de décomposition finie par distance.



Un résultat majeur, établi depuis plus de vingt ans, est que les graphes des automates à pile sont les graphes de degré fini et admettant une décomposition finie par distance.

Théorème 4.1.3. *Muller et Schupp [MS85] Les graphes de transition des automates à pile sont les graphes de degré fini, accessibles à partir d'un unique sommet initial et de décomposition finie.*

Démonstration :
$\Rightarrow$: nous montrons d'abord que le graphe de transition de tout automate à pile est un graphe de degré et de décomposition finis. Soit $M = (S, \Gamma, \Sigma, s_0, F, \delta)$ un automate à pile.
Par définition, son graphe de transition G_M a un unique sommet initial $s_0\perp$ qui est racine de G_M.
Vérifions que G_M et plus généralement

$$H_M \;=\; \{pAU \xrightarrow{a} qVU \mid pA \xrightarrow{a} qV \in \delta \wedge AU \in \Gamma^*\perp\}$$

est un graphe de degré fini.
Le degré sortant de tout sommet (pAU) est égal au nombre de règles de domaine pA, donc est borné par le nombre $|\delta|$ de règles. Vérifions que le

degré entrant de tout sommet (qW) est fini.
Supposons qu'il existe une infinité d'arcs de but (qW) :
pour tout $n \geq 0$, $p_nA_nU_n \underset{H_M}{\overset{a_n}{\rightarrow}} qW$ avec $(p_iA_iU_i, a_i) \neq (p_jA_jU_j, a_j)$ pour $i \neq j$.
Pour tout $n \geq 0$, soit q_nV_n le membre droit de la règle (p_nA_n, a_n) *i.e.*

$$p_nA_n \overset{a_n}{\rightarrow} q_nV_n \ \in \ \delta.$$

Ainsi, on a

$$qW \ = \ q_1V_1U_1 \ = \ \ldots \ = \ q_nV_nU_n \ = \ \ldots$$

d'où $q = q_1 = \ldots = q_n = \ldots$ et $W = V_1U_1 = \ldots = V_nU_n = \ldots$
Comme la relation δ est finie, il existe $i \neq j$ tels que

$$(p_i, A_i, a_i, q_i, V_i) = (p_j, A_j, a_j, q_j, V_j).$$

Comme $V_i = V_j$, on a $U_i = U_j$ d'où $(p_iA_iU_i, a_i) = (p_jA_jU_j, a_j)$, ce qui est contradictoire. Donc G_M est de degré fini.
On montre que G_M est de décomposition finie par distance en deux étapes. La première étape montre l'identité entre la décomposition finie par distance et le fait d'être engendré par distance par une grammaire (voir proposition 4.2.4). La deuxième étape montre que tout graphe régulier de degré fini et accessible est finiment décomposable par distance ([Ca07]).
$\Leftarrow$: soit G un graphe de degré fini, ayant un unique sommet initial r qui est racine de G, et de décomposition finie par distance. Montrons que G est isomorphe au graphe de transition d'un automate à pile. On note $Comp(H)$ l'ensemble des composantes connexes d'un graphe H.
Soient $G_1, \ldots G_p$ des graphes non isomorphes tels que

$$\forall j \in [p], \ \exists n \geq 0, \ \exists H \in Comp(\widehat{G}^n), \ G_j \simeq H$$
$$\forall n \geq 0, \ \forall H \in Comp(\widehat{G}^n), \ \exists j \in [p], \ G_j \simeq H.$$

On suppose que les sommets initiaux des graphes G_i sont distincts entre eux :

$$\mathcal{S}_{G_k,i} \cap \mathcal{S}_{G_l,i} = \emptyset \qquad \text{pour tout } 1 \leq k \leq l \leq p.$$

Pour tout $j \in [p]$, il existe des applications injectives $f^1_{j,1}, \ldots, f^{n_1}_{j,1}, \ldots, f^1_{j,p}, \ldots, f^{n_p}_{j,p}$ telles que

$$\begin{array}{rcl} \widehat{G_j} & = & f^1_{j,1}(G_1) \cup \ldots \cup f^{n_1}_{j,1}(G_1) \cup \ldots \\ & \cup & f^1_{j,p}(G_p) \cup \ldots \cup f^{n_p}_{j,p}(G_p) \end{array}$$

et où les unions sont disjointes.
On construit un automate à pile dont l'ensemble S des états est

$$S = \mathcal{S}_{G_1,i} \cup \ldots \cup \mathcal{S}_{G_p,i} \ ,$$

l'alphabet Γ des lettres de pile ($\perp \notin \Gamma$) est

$$\Gamma = \{(j,k,l) \mid j,k \in [p] \ \wedge \ l \in [n_k]\}$$

et la relation de transition δ est définie par les règles ci-dessous.
On a les règles d'ajout :

$$p(x,j,y) \xrightarrow{a} q(j,k,m)(x,j,y) \qquad \text{si } p \xrightarrow[G_j]{a} f^m_{j,k}(q)$$

les règles de retrait :

$$q(j,k,m) \xrightarrow{a} p \qquad \text{si } f^m_{j,k}(q) \xrightarrow[G_j]{a} p$$

et les règles d'action interne :

$$p(x,j,y) \xrightarrow{a} q(x,j,y) \qquad \text{si } p \xrightarrow[G_j]{a} q.$$

On suppose que $G_1 = G$ et on ajoute les règles :

$$\begin{array}{lll} r\perp \xrightarrow{a} q(1,k,m)\perp & \text{si} & r \xrightarrow[G_1]{a} f^m_{1,k}(q) \\ q(1,k,m)\perp \xrightarrow{a} r\perp & \text{si} & f^m_{1,k}(q) \xrightarrow[G_1]{a} r \\ r\perp \xrightarrow{a} r\perp & \text{si} & r \xrightarrow[G_1]{a} r. \end{array}$$

L'état initial est r et l'ensemble des états finaux est

$$F \ = \ \{s \in \mathcal{S}_{G_j,i} \mid j \in [p] \wedge fs \in G_j\}.$$

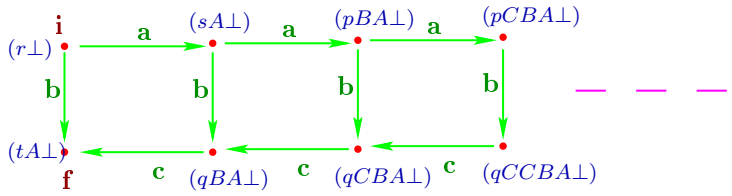
Ainsi, le graphe de transition de cet automate à pile, accessible à partir de $r\perp$, est isomorphe au graphe G.

□

La construction de ce théorème appliquée à l'exemple précédent donne l'automate à pile M suivant :

$$\begin{array}{l|l|l} r\bot \xrightarrow{a} sA\bot & sA \xrightarrow{a} pBA & pB \xrightarrow{a} pCB \\ r\bot \xrightarrow{b} tA\bot & sA \xrightarrow{b} qBA & pB \xrightarrow{b} qCB \\ qB \xrightarrow{c} t & pC \xrightarrow{a} pCC & \\ qC \xrightarrow{c} q & pC \xrightarrow{b} qCC & \end{array}$$

avec $A = (1, 2, 1)$, $B = (2, 3, 1)$, $C = (3, 3, 1)$, d'état initial r et d'unique état final t. Le graphe de transition de M est le graphe



Suivant cette idée, il a été développé un outil pour engendrer ces graphes, à savoir les grammaires HR (*Hyperedge Replacement*) [Co90] dites aussi grammaires déterministes de graphes.

4.2 Automates réguliers

Les grammaires algébriques et les automates à pile sont des machines abstraites pour engendrer ou reconnaître les langages algébriques. Le théorème de Muller et Schupp a établi que les graphes des automates à pile sont les graphes ayant une régularité de structure : ils admettent une décomposition finie par distance à partir des sommets initiaux. Au lieu de partir du graphe d'un automate à pile et de le découper itérativement, on préfère l'engendrer itérativement à partir du graphe vide (en ajoutant des sommets au lieu d'en enlever) à l'aide d'une grammaire déterministe de graphes.

4.2.1 Les grammaires déterministes de graphes

On étend la notion de grammaire de mots en la notion de grammaire de graphes. Là où une grammaire de mots engendre un langage, une grammaire

déterministe de graphes engendre un graphe. Néanmoins, le principe qui régit la construction du langage dans une grammaire de mots, la récriture, s'étend aux grammaires de graphes.
Pour définir une grammaire de graphes, nous prenons un alphabet $\mathcal{F}$ de symboles, appelés fonctions, munis d'une arité ρ.
Un *hyperarc* est un arc portant sur plusieurs sommets. Il est étiqueté par une fonction ayant pour arité le nombre de sommets sur lesquels l'hyperarc porte. L'hyperarc $fs_1 \dots s_{\rho(f)}$ est étiqueté par f, de sommets successifs $s_1, \dots, s_{\rho(f)}$. De plus, on divise $\mathcal{F}$ en deux sous-ensembles distincts $\mathcal{F} = N \uplus T$. Les éléments de N sont appelés les *non-terminaux* et ceux de T les *terminaux*. Pour tout $n \geq 0$,

$$\mathcal{F}_n \;=\; N_n \uplus T_n$$

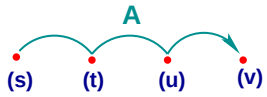
où N_n et T_n sont respectivement les non-terminaux et les terminaux d'arité n. On étend la notion d'arité des symboles à l'arité des hyperarcs et on dit que $fs_1 \dots s_{\rho(f)}$ est d'arité $\rho(f)$:

$$\rho(fs_1 \dots s_{\rho(f)}) \;=\; \rho(f) \qquad \text{pour tout } f \in \mathcal{F}.$$

Un hyperarc $fs_1 \dots s_{\rho(f)}$ est qualifié de non-terminal (respectivement terminal) si f est un non-terminal (resp. un terminal). On fera également référence à un hyperarc vu comme un mot de $\mathcal{F}\mathcal{S}^*$ pour un ensemble $\mathcal{S}$ de sommets. Notamment, la première lettre d'un hyperarc est son étiquette :

$$(fs_1 \dots s_{\rho(f)})(1) \;=\; f.$$

La figure suivante présente l'hyperarc *Astuv* d'arité 4.



Un *hypergraphe* G est un sous-ensemble de $\bigcup_{n\geq 0} \mathcal{F}_n\mathcal{S}^n$ pour un ensemble $\mathcal{S}$ de sommets. L'ensemble des sommets de G est

$$\mathcal{S}_G \;:=\; \{s \in \mathcal{S} \mid \mathcal{F}\mathcal{S}^* s \mathcal{S}^* \cap G \neq \emptyset\}.$$

Prenant un ensemble E d'étiquettes, on définit l'ensemble $\mathcal{S}_{G,E}$ des sommets de G qui sont sommets des hyperarcs étiquetés par un symbole de E :

$$\mathcal{S}_{G,E} \ := \ \{s \in \mathcal{S}_G \mid E\mathcal{S}^* s \mathcal{S}^* \cap G \neq \emptyset\} \ = \ \mathcal{S}_{G \cap E\mathcal{S}^*}$$

où $G \cap E\mathcal{S}^*$ est la restriction de G aux hyperarcs étiquetés dans E.

Pour faire le parallèle avec les notions d'étiquettes et de couleurs du paragraphe 1.4, nous utiliserons le terme « étiquette d'arcs »pour désigner les terminaux d'arité 2 et le terme « couleur » pour désigner les terminaux d'arité 1.

Une *grammaire de graphes* $R = (\Sigma, N, D, \delta)$ est composée de :
- un ensemble fini Σ d'étiquettes,
- un ensemble fini N de symboles non-terminaux,
- un hypergraphe fini D dit axiome de la grammaire, et
- un ensemble fini δ de couples tel que $\delta \subseteq N\mathcal{S}^* \times 2^{(\Sigma \cup N)\mathcal{S}^*}$.

L'ensemble δ des règles de la grammaire est un ensemble de couples pour lesquels la première composante (le membre gauche) est un hyperarc non-terminal reliant des sommets distincts et la deuxième composante (le membre droit) est un hypergraphe fini.
On se restreint aux *grammaires déterministes de graphes*, c'est-à-dire qu'il n'y a qu'une seule règle pour chaque non-terminal :

$$(X, H), (Y, K) \in \delta \wedge X(1) = Y(1) \implies (X, H) = (Y, K).$$

Désignons par
- N_R l'ensemble des étiquettes des membres gauches de δ, aussi appelées les *non-terminaux* de R :
$$N_R \ := \ \{f \in N \mid \exists (X, Y) \in \delta, \ X(1) = f\}$$
- T_R l'ensemble des *terminaux* apparaissant dans les membres droits de R :
$$T_R \ := \ \{f \in T \mid \exists (X, Y) \in \delta \ \exists s_1 \dots s_{\rho(f)}, \ f s_1 \dots s_{\rho(f)} \in Y\}$$
- F_R l'ensemble des étiquettes de R :
$$F_R \ := \ N_R \cup T_R.$$

Pour notre étude, nous ne considérons que des terminaux d'arité un ou deux. En effet, nous utilisons les grammaires comme générateurs de graphes étiquetés et coloriés : les terminaux d'arité 1 sont les couleurs des sommets et ceux d'arité 2 étiquettent les arcs. Ainsi $T_R \subset \mathcal{F}_1 \cup \mathcal{F}_2$.
Pour toute règle $(X, Y) \in \delta$, on spécifie l'ensemble I_Y des *entrées* qui sont les sommets de Y qui apparaissent dans X, et l'ensemble O_Y des *sorties* de Y qui sont les sommets qui appartiennent à un hyperarc non-terminal dans Y :

$$I_Y := \mathcal{S}_X \cap \mathcal{S}_Y \qquad \text{et} \qquad O_Y := \mathcal{S}_{Y,N_R}$$

Comme pour les grammaires de mots, on donne un axiome, D, qui est un point de départ pour la grammaire. Dans le cas des grammaires de graphes, D est un hypergraphe fini. Pour toute grammaire R, on pose $Z \in N_0 - F_R$ un non-terminal particulier n'apparaissant pas dans la grammaire et on ajoute à R la règle (Z, D). Une grammaire de graphes se résumera alors au triplet (Σ, N, R) et plus généralement, on omet les ensembles d'étiquettes et une grammaire R se résume à son ensemble de règles. Ainsi, $Dom(R)$ (resp. $Im(R)$) est l'ensemble des membres gauches (resp. droits) des règles de R.

Exemple 4.2.1. Soit la grammaire de graphes

$$R = \{(Z, \{Axy, ix, fy\}), (A12, \{a1p, bp2, Apq, bq2\})\}$$

avec $N_R = \{Z, A\}$ et $T_R = \{a, b, i, f\}$. La forme littéraire pouvant être difficile à lire, on a représenté R à la figure 4.6.

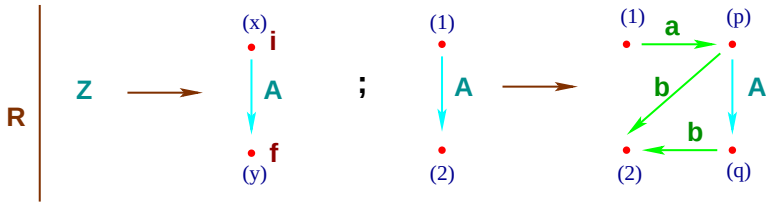


FIG. 4.6 – Une grammaire déterministe de graphes.

Nous avons vu précédemment le fonctionnement des grammaires de mots. Nous étendons le principe de récriture aux grammaires de graphes. Pour les

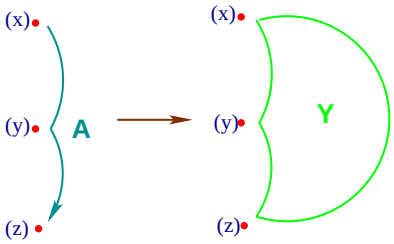
mots, on remplace un non-terminal par un mot. Dans le cas des graphes, un hyperarc non-terminal P d'un hypergraphe H est remplacé par le membre droit d'une règle (X, Y) en identifiant les entrées de Y avec les sommets correspondants de P, la correspondance ne se faisant que si P et X ont la même étiquette. Un hypergraphe H *se récrit* en un hypergraphe H' en remplaçant un hyperarc non-terminal P suivant une grammaire R , et on note $H \underset{R,P}{\longrightarrow} H'$, si

$$\exists\ (X, Y) \in R,\ X(1) = P(1)\ \wedge\ H' = (H - P) \cup h(Y)$$

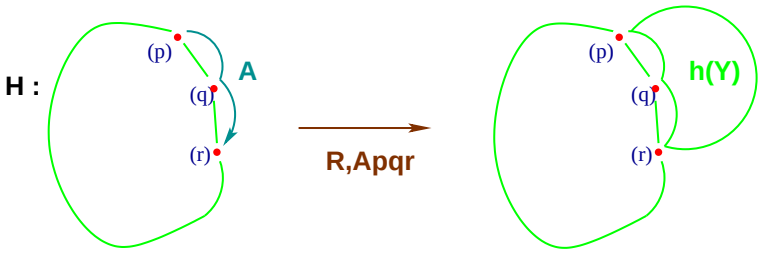
où h est une application qui à toute entrée $X(i)$ associe le sommet $P(i)$ (pour $i \in [2, \rho(X(1))]$), et associe injectivement aux autres sommets de Y de nouveaux éléments de $\mathcal{S}$:

$$\begin{array}{rcccl} h & : & X(i) \in I_Y & \mapsto & P(i) \qquad \text{pour tout } i \in [2, \rho(X(1))]. \\ & & \mathcal{S}_Y - I_Y & \hookrightarrow & \mathcal{S} - \mathcal{S}_H \end{array}$$

Pour une règle de R représentée par



la récriture d'un hyperarc *Apqr* d'un hypergraphe H est représenté par



avec $h(x) = p$, $h(y) = q$, $h(z) = r$ et $h(s) \notin \mathcal{S}_H$ pour tout $s \in \mathcal{S}_Y - \{x, y, z\}$.

On représente à la figure 4.7 les premières étapes de récriture en fonction de la grammaire de l'exemple 4.2.1.

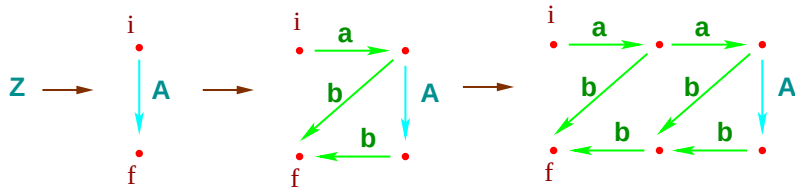


FIG. 4.7 – Récritures de graphes.

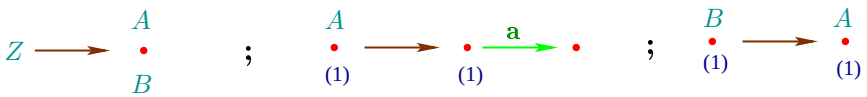
La récriture $\underset{R,P}{\longrightarrow}$ d'un hyperarc P selon une grammaire R est étendue à la récriture $\underset{R,Q}{\longrightarrow}$ d'un ensemble Q d'hyperarcs non-terminaux selon R. En particulier, la *récriture parallèle complète* $\underset{R}{\Longrightarrow}$ est la récriture de tous les hyperarcs non-terminaux d'un hypergraphe :

$$H \underset{R}{\Longrightarrow} H' \qquad \text{si} \qquad H \underset{R,H\cap NS^*_H}{\longrightarrow} H'.$$

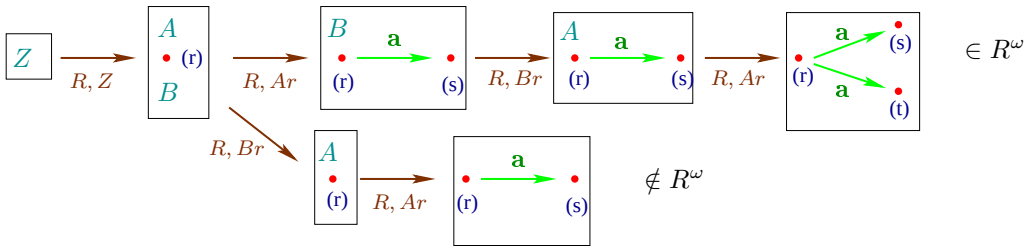
Pour un hypergraphe H, on note $[H]$ la restriction aux arcs terminaux et aux sommets coloriés : $[H] := H - NS^*_H$.
Une grammaire (déterministe de graphes) R engendre, à partir de tout hypergraphe H étiqueté dans $N_R \cup \mathcal{F}_1 \cup \mathcal{F}_2$ et en itérant la récriture parallèle, une infinité de graphes mais tous isomorphes :

$$R^\omega(H) \;=\; \{ \bigcup_{n\geq 0} [H_n] \mid H = H_0 \underset{R}{\Rightarrow} \dots H_n \underset{R}{\Rightarrow} H_{n+1} \dots \}$$

En particulier, l'ensemble $R^\omega(Z)$ (ou plus simplement R^ω) des *graphes engendrés* par R est la fermeture par isomorphisme d'un graphe engendré. On parlera aussi du graphe engendré par R (à isomorphisme près).
Remarquons que à chaque étape de récriture parallèle, le nombre de sommets nouvellement atteint est fini. On peut remarquer également que dans la définition de R^ω, la récriture parallèle est utile afin d'éviter qu'un hyperarc non-terminal ne soit pas récrit mais aussi afin d'éviter la non confluence de la récriture liée à la non multiplicité des graphes, comme le montre l'exemple suivant (les graphes obtenus par récriture sont ceux encadrés). Pour la grammaire R ci-dessous



on a



On peut également définir le *graphe canonique* $Gen(R)$ d'une grammaire R : chaque sommet s de $Gen(R)$ est un mot de $N^*\mathcal{S}_{\cup Im(R)}$ qui est la suite de non-terminaux et le sommet non-entrée utilisés pour accéder à s par récritures à partir de Z. Nous renvoyons le lecteur au chapitre 3.5 de [Ca07] pour une construction détaillée.

Un graphe G est *régulier* si il existe une grammaire de graphes qui l'engendre.
A la figure suivante, on représente le graphe régulier engendré par la grammaire de l'exemple 4.2.1.

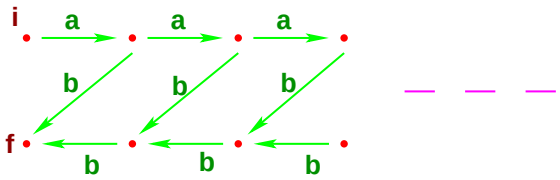


FIG. 4.8 – Un graphe régulier.

Deux grammaires R et S sont *équivalentes* si elles engendrent le même graphe : $R^\omega = S^\omega$.

Pour un graphe régulier G engendré par une grammaire de graphes R, on définit le *niveau* $l^R_G(s)$ d'un sommet s comme étant le nombre minimum de récritures de R à partir de son axiome nécessaires pour atteindre le sommet s, à savoir pour $G = \underset{n\geq 0}{\cup}[H_n]$ avec $Z \underset{R}{\Rightarrow} H_0 \ldots \underset{R}{\Rightarrow} H_n \underset{R}{\Rightarrow} \ldots$,

$$l^R_G(s) := min\{n \mid s \in \mathcal{S}_{H_n}\}.$$

En particulier, les sommets de l'axiome sont de poids 0. On pourra utiliser la notation simplifiée $l_G(s)$ si la grammaire engendrant le graphe G est entendue. On représente à la figure 4.9 le niveau des sommets du graphe engendré par la grammaire 4.6.

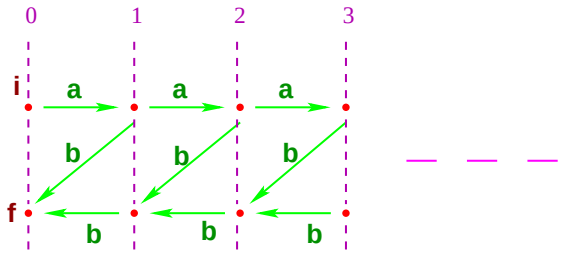


FIG. 4.9 – Niveau des sommets du graphe engendré par la grammaire 4.6.

Les grammaires de graphes peuvent être normalisées de différentes manières.

On considère que toute grammaire est *terminale aux sorties* (TS). Dans tout membre droit, chaque hyperarc terminal relie au moins un sommet qui n'est pas une entrée :

$$\forall (X,Y) \in R,\ \forall f s_1 \dots s_{\rho(f)} \in Y \cap T_R S^* \implies \{s_1, \dots, s_{\rho(f)}\} \cap (\mathcal{S}_Y - I_X) \neq \emptyset$$

en particulier, toute couleur ne porte pas sur une entrée. La grammaire présentée à la figure 4.6 est terminale aux sorties.
L'intérêt d'une grammaire R d'être TS est que pour $G = \bigcup_{n\geq 0} [H_n]$ avec $Z \underset{R}{\Rightarrow} H_0 \dots \underset{R}{\Rightarrow} H_n \underset{R}{\Rightarrow} \dots$, on a pour tout $n \geq 0$,

$$[H_n] \ = \ G_{|\{s \ \mid \ l(s)\leq n\}}.$$

Ceci n'est pas une restriction : toute grammaire peut être transformée en une grammaire TS équivalente.

Lemme 4.2.2. *Toute grammaire R peut être transformée en une grammaire TS équivalente S préservant les niveaux : pour tout $G \in R^\omega$,*

$$l_G^R(s) \ = \ l_G^S(s) \qquad \textit{pour tout } s \in \mathcal{S}_G.$$

Démonstration :
Pour tout $X \in Dom(R)$, on détermine le graphe fini

$$H_X \;=\; G \cap T_R \mathcal{S}_X^* \qquad \text{pour tout } G \in R^\omega(X)\,;$$

en particulier $R(X) \cap T_R \mathcal{S}_X^* \subseteq H_X$.
On définit la grammaire suivante :

$$I \;=\; \{(X, H_X \cup \{X\}) \mid X \in Dom(R)\}$$

et pour tout $X \in Dom(R)$, on associe le graphe

$$K_X \;:=\; H - H_X \text{ tel que } R(X) \underset{I}{\Rightarrow} H.$$

Il suffit alors de prendre la grammaire

$$S \;:=\; \{(X, K_X) \mid X \in Dom(R)\}$$

qui est terminale aux sorties. Par construction, $R^\omega \;=\; S^\omega$ et pour tout graphe $G \in R^\omega$, $l_G^R(s) \;=\; l_G^S(s)$ pour tout sommet s de G.

□

La mise en forme terminale aux sorties de la grammaire à la figure 4.10 est illustrée à la figure 4.11. Cette grammaire est TS et équivalente, et les grammaires définissent la même relation de niveau.

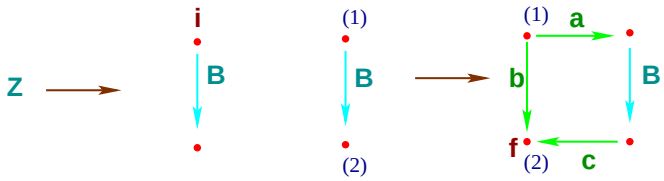


FIG. 4.10 – Une grammaire non terminale aux sorties.

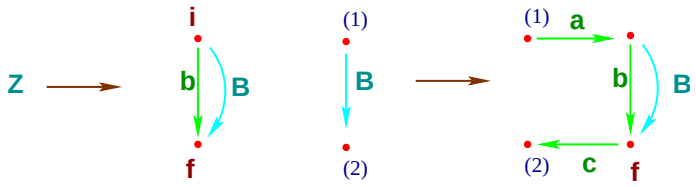


FIG. 4.11 – Mise sous forme terminale aux sorties (TS).

Dorénavant, toute grammaire est supposée être TS.

Une autre normalisation nous permet de ne considérer que des grammaires connexes. Une grammaire R est une *grammaire connexe* si de tout membre gauche qui n'est pas une constante, R engendre un graphe connexe :

$$\forall X \in Dom(R) - F_0,\ \forall G \in R^\omega(X),\ G \text{ est connexe.}$$

Toute grammaire peut être mise sous forme connexe.

Lemme 4.2.3. *Toute grammaire R peut être transformée en une grammaire équivalente S connexe et préservant les niveaux.*

Démonstration :
A toute règle $Ax_1 \dots x_{\rho(A)} \to H_A$ de R et pour tout $1 \leq i \leq \rho(A)$, on associe l'ensemble $Con(A,i)$ des sommets de H_A connectés au sommet x_i dans le graphe $R^\omega(Ax_1 \dots x_{\rho(A)})$. L'ensemble des $Con(A,i)$ est le plus petit point fixe du système suivant :

$$\begin{aligned} Con(A,i) &= \{x_i\} \cup \bigcup\{\mathcal{S}_X \mid X \in H_A \wedge X(1) \in T_R \wedge \mathcal{S}_X \cap Con(A,i) \neq \emptyset\} \\ &\cup \bigcup\{U(j) \mid \exists Y \in Dom(R),\ Y(1)U \in H_A \\ &\qquad \wedge \exists k,\ U(k) \in Con(A,i) \wedge x_j \in Con(Y(1),k)\}. \end{aligned}$$

On complète ces ensembles en définissant, pour tout non-terminal $A \in N_R$, l'ensemble

$$Con(A) := \{Con(A,i) \mid 1 \leq i \leq \rho(A)\} \cup \{\emptyset\}.$$

Pour chaque hyperarc non-terminal $X \in Dom(R)$ et chaque $P \in Con(X(1))$, on associe un nouveau symbole $X(1)_P$ d'arité $|P \cap \{x_1, \ldots, x_{\rho(X(1))}\}|$, et l'hyperarc

$$X_P \ := \ X(1)_P x_{i_1} \ldots x_{i_p}$$

avec $\{x_{i_1}, \ldots, x_{i_p}\} = P \cap \{x_1, \ldots, x_{\rho(X(1))}\}$ et $i_1 < \ldots < i_p$. En particulier $X_\emptyset \ = \ X(1)_\emptyset$ est une constante.
On définit alors la grammaire suivante :

$$I \ := \ \{(X, \{X_P \mid P \in Con(X(1))\}) \mid X \in Dom(R)\}$$

ce qui découpe chaque $X \in Dom(R)$ en hyperarcs en fonction de $Con(X(1))$. Pour chaque règle (X, H) de R, il y a un unique hypergraphe K_X tel que $H \underset{I}{\Rightarrow} K_X$, et on note

$$\ll X \gg := \ \mathcal{S}_{K_X} - \bigcup Con(X(1))$$

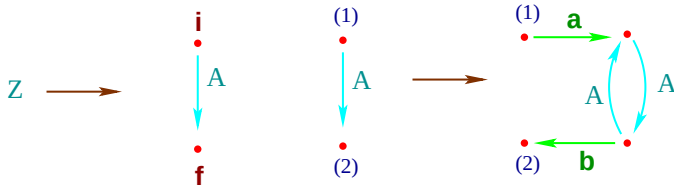
l'ensemble des sommets de H_X qui ne sont pas reliés à un sommet de $\mathcal{S}_X$. La grammaire suivante :

$$\begin{array}{rcl} S & := & \{(X_P, (K_X)_{|P}) \mid X \in Dom(R) \wedge P \in Con(X(1)) - \{\emptyset\}\} \\ & \cup & \{X_\emptyset, (K_X)_{|\ll X \gg} \mid X \in Dom(R)\} \end{array}$$

est connexe. De plus, $R^\omega = S^\omega$ et pour tout $G \in R^\omega$, $l^R_G(s) \ = \ l^S_G(s)$ pour tout sommet s de G.

□

La construction du lemme 4.2.3 appliquée à la grammaire suivante :



donne la grammaire connexe équivalente et préservant les niveaux :

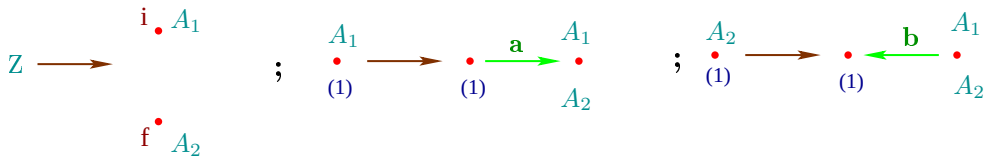


FIG. 4.12 – Mise sous forme connexe de grammaire.

Engendrer un graphe par une grammaire est dual de le décomposer finiment : au lieu de partir de tout le graphe, on commence par le graphe vide (c'est-à-dire du non-terminal Z) et on ajoute itérativement par récriture parallèle ce que l'on enlève par décomposition. Ainsi, décomposer finiment par distance un graphe de degré fini correspond à l'engendrer par une grammaire de graphes, de sorte que le niveau de tout sommet du graphe est sa distance.

Proposition 4.2.4. *Pour tout graphe G de degré fini et pour tout $P \subseteq \mathcal{S}_G$ fini,*

$$\begin{array}{ll} & G \text{ est de décomposition finie par distance à partir de } P \\ \iff & G \text{ est engendré par une grammaire } R \text{ telle que } d_G(s,P) = l_G^R(s) \\ & \text{pour tout } s \in \mathcal{S}_G. \end{array}$$

Démonstration :

$\Longrightarrow$: soit G un graphe de degré fini de décomposition finie par distance à partir de $P \subseteq \mathcal{S}_G$. On suppose que i ne colore pas G et on note $\overline{G} := G \cup \{iP\}$ de sorte que

$$d_G(s,P) = d_{\overline{G}}(s) = min\{d_G(s,t) \mid t \in P\}.$$

Soient $G_1, \ldots, G_p$ des graphes non isomorphes représentant les composantes connexes de la décomposition par distance de $\overline{G}$:

$$\begin{array}{l} \forall j \in [p]\ \exists n \geq 0\ \exists H \in Comp(\widehat{\overline{G}}^n)\ G_j \simeq H \\ \forall n \geq 0\ \forall H \in Comp(\widehat{\overline{G}}^n)\ \exists j \in [p]\ G_j \simeq H. \end{array}$$

Pour tout $j \in [p]$, on note r_j le nombre de sommets de G_j coloriés par i :

$$r_j := |G_j \cap i\mathcal{S}_{G_j}|$$

et on suppose que ses sommets sont $1, \ldots, r_j$:

$$G_j \cap i\mathcal{S}_{G_j} \;=\; \{i1, \ldots, ir_j\}.$$

A tout $j \in [p]$, on associe un nouveau symbole $[j]$ d'arité r_j et la règle suivante :

$$\begin{array}{rcl} [j]1 \ldots r_j & \rightarrow & (G_j - i\mathcal{S}_{G_j}) - \widehat{G_j} \\ & & \cup\{[k]f(1) \ldots f(r_K) \mid \exists H \in Comp(\widehat{G_j}),\ H = f(G_K) \\ & & \qquad\qquad \wedge\ f \text{ injective } \}. \end{array}$$

On suppose que G_1 est isomorphe à G. On ajoute aux règles précèdentes la règle $Z \rightarrow [1]1 \ldots r_1$ pour obtenir une grammaire R engendrant G par distance :

$$l_G^R(s) \;=\; d_G(s, P) \text{ pour tout } s \in \mathcal{S}_G.$$

Par le lemme 4.2.2, on transforme R en une grammaire TS équivalente et préservant les niveaux des sommets.
$\Longleftarrow$: soit R une grammaire engendrant par distance croissante. Soit $G \in R^\omega$. On a

$$l_G^R(s) \;=\; d_G(s, P) \text{ pour tout sommet } s \text{ de } G$$

où $P = \{s \in \mathcal{S}_G \mid l_G^R(s) = 0\}$. D'après le lemme 4.2.3, on peut supposer que R est connexe. Comme G est connexe à partir de P, $N_R \cap F_0 = \{Z\}$. On considère l'ensemble fini suivant :

$$E \;:=\; \bigcup\{Comp(H_{|\mathcal{S}_H - \mathcal{S}_X}) \mid (X, H) \in R\}.$$

Pour toute composante connexe H obtenue par décomposition par distance de G à partir de P, il existe $K \in E$ tel que $H \in R^\omega(K)$. Comme E est fini, G est de décomposition finie par distance à partir de P.

□

Nous avions déjà indiqué que les automates à pile et les grammaires de graphes avaient la même expressivité en termes de langages. Nous donnons ici une construction permettant de transformer un automate à pile en grammaire de graphes, de sorte que le graphe de transition de l'automate à pile est régulier par longueur croissante.

Proposition 4.2.5. *On peut transformer tout automate à pile M en une grammaire de graphes R de sorte que $G_M \in R^\omega$ et $|s| = l^R_G(s) + 2$ pour tout $s \in \mathcal{S}_{G_M}$.*

Démonstration :
Soit $M = (S, \Gamma, \Sigma, s_0, F, \delta)$ un automate à pile. On ordonne les états de M : $S = \{p_1, \dots p_m\}$. A toute lettre de pile $A \in \Gamma$, on associe un symbole $A \in \mathcal{F}_M$ d'arité m. On définit la grammaire (déterministe de graphes) R suivante :

$$R = \{(Z, H_\perp \perp)\} \cup \{(Ap_1 \dots p_m, H_A) \mid A \in \Gamma\}$$

où pour tout $A \in \Gamma \cup \{\perp\}$,

$$\begin{array}{rcll} H_A & = & \{p \xrightarrow{a} q \mid pA \xrightarrow[\delta]{a} qA\} & \text{action interne} \\ & \cup & \{p \xrightarrow{a} qB \mid pA \xrightarrow[\delta]{a} qBA\} & \text{ajout} \\ & \cup & \{pB \xrightarrow{a} q \mid pB \xrightarrow[\delta]{a} q\} & \text{retrait} \\ & \cup & \{B(p_1B) \dots (p_mB) \mid B \in \Gamma\} & \text{propagation} \\ & \cup & \{is_0 \mid A = \perp\} & \text{état initial} \\ & \cup & \{fp \mid p \in F\} & \text{états finaux.} \end{array}$$

Par construction, $H_M \in R^\omega$ et $|s| = l^R_{H_M}(s) + 2$ pour tout $s \in \mathcal{S}_{H_M}$. Il ne reste plus qu'à appliquer le lemme 4.2.2 pour que R soit bien formé (TS) puis la propriété 4.2.6 pour engendrer $G_M = (H_M)_{\downarrow s_0 \perp}$.

□

La construction de la proposition 4.2.5 appliquée à l'automate à pile de l'exemple 4.1.1 donne la grammaire de la figure 4.13 (en forme TS et accessible à partir de i) Nous disposons donc d'un outil, les grammaires de graphes, ayant autant de pouvoir d'expression que les automates à pile. En utilisant ces grammaires, on étend la notion d'automate fini à celle d'automate régulier, nous permettant de définir le langage d'une grammaire.

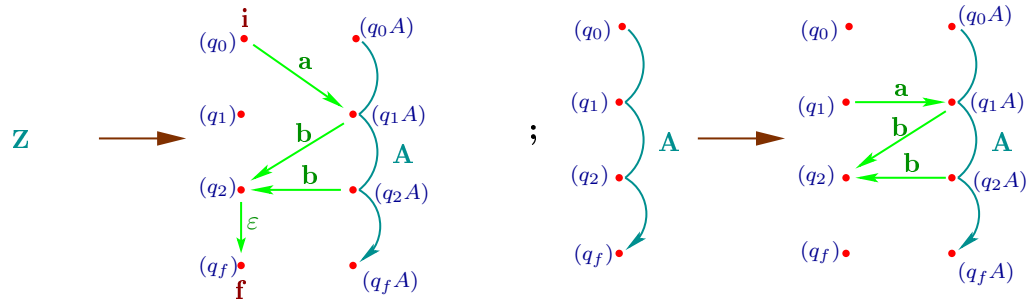


FIG. 4.13 – Transformation d'un automate à pile en grammaire de graphes.

4.2.2 Les automates réguliers

Comme pour les automates finis, on prend deux couleurs particulières, i et f, afin de distinguer respectivement les états initiaux et les états finaux d'un automate. Un *automate régulier* $G = (\mathcal{S}, \mathcal{A})$ est un graphe régulier ; on note I l'ensemble de ses sommets initiaux et F l'ensemble de ses sommets finaux :

$$I \;:=\; \mathcal{S}_{G,i} \qquad \text{et} \qquad F \;:=\; \mathcal{S}_{G,f}.$$

Le langage $L(G)$ *accepté* par un automate régulier G est défini comme étant l'ensemble des traces de G d'un sommet initial à un sommet final :

$$L(G) \;:=\; L(G, I, F).$$

Nous distinguons également le *langage initial* $L(G, i)$ de G qui est le langage des traces de G d'un sommet initial à n'importe quel sommet de G :

$$L(G, i) \;:=\; L(G, I, \mathcal{S}_G).$$

De plus, si G est engendré par une grammaire de graphes R, on dit que le langage $L(R)$ *accepté* par la grammaire R est le langage accepté par G, de même pour le *langage initial* $L(R, i)$ de R :

$$L(R) \;=\; L(G) \qquad \text{et} \qquad L(R, i) \;=\; L(G, i).$$

Pour la grammaire de la figure 4.6, on a $L(R) = \{a^n b^n \mid n \geq 0\}$ et $L(R, i) = \{a^m b^n \mid 0 \leq n \leq m\}$.

Par la suite, nous ne considérerons que des automates réguliers dont les sommets initiaux n'apparaissent que dans l'axiome des grammaires qui les engendrent. Nous montrerons (proposition 5.3.2) que ce n'est pas une restriction pour notre étude.
Pour toute grammaire R engendrant un graphe G, il existe une grammaire engendrant la restriction de G aux sommets accessibles à partir de i et préservant le niveau des sommets.

Propriété 4.2.6. *On peut transformer toute grammaire R en une grammaire S telle que pour tout $G \in R^\omega$, le graphe $H = G_{i\rightarrow} \in \mathcal{S}^\omega$ et $l^R_G(s) = l^S_H(s)$ pour $s \in G_{i\rightarrow}$.*

Démonstration :
Soit R une grammaire engendrant le graphe G. Nous construisons une grammaire R' telle que $G_{i\rightarrow} \in R'^\omega$.
Au renommage de sommets près, on considère que les membres gauches des règles de R sont de la forme $A1 \dots \rho(A)$ où $A \in N_R$.
Pour chaque règle $A1 \dots \rho(A) \rightarrow H_A$ de R et chaque $I \subseteq [\rho(A)]$, on associe l'ensemble $Acc(A, I)$ des sommets de H_A accessibles à partir d'un sommet de I dans le(s) graphe(s) $R^\omega(A1 \dots \rho(A))$. L'ensemble des $Acc(A, I)$ est le plus petit point fixe du système suivant :

$$\begin{aligned} Acc(A,I) \;:=\; & I \cup \{s \mid is \in H_A\} \\ \cup\; & \{s \mid \exists t \in Acc(A,I),\ \exists a,\ t \xrightarrow{a} s \in H_A\} \\ \cup\; & \{Y(i) \mid \exists B \in N_R,\ BY \in H_A \\ & \qquad \wedge\, i \in Acc(B, \{j \mid Y(j) \in Acc(A,I)\})\}. \end{aligned}$$

A chaque couple (A, I) tel que $I = Acc(A, I) \cap [\rho(A)]$, on associe un nouveau non-terminal A_I d'arité $|I|$ et on définit la règle suivante

$$A_I j_1 \dots j_{|I|} \longrightarrow H_{A,I} \qquad \text{avec } j_1 < \dots < j_{|I|} \text{ et } \{j_1, \dots, j_{|I|}\} = I$$

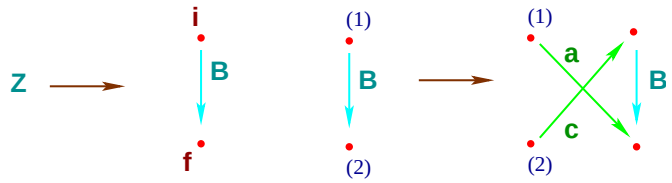
et où

$$\begin{aligned} H_{A,I} \;:=\; & [R(A)]_{|Acc(A,I)} \\ \cup\; & \{B_J Y(b_1) \dots Y(b_n) \mid BY \in H_A \wedge B \in N_R \wedge b_1 < \dots < b_n \\ & \qquad \wedge\, J = \{b_1, \dots, b_n\} = Acc(A,I) \cap \{Y(1), \dots, Y(|Y|)\}\}. \end{aligned}$$

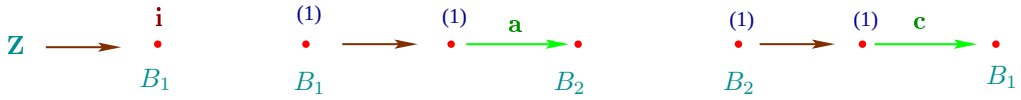
Par restriction aux non-terminaux accessibles à partir de Z, on obtient la grammaire R' souhaitée.

□

En appliquant cette construction à la grammaire suivante,



on obtient la grammaire présentée ci-après



Pour toute grammaire R, on note

$$R^{\omega}_{i\rightarrow} \;:=\; \{G_{i\rightarrow} \mid G \in R^{\omega}\}$$

la restriction des graphes engendrés aux sommets accessibles à partir des sommets initiaux.
L'*inverse* d'une grammaire R est une grammaire R^{-1} construite à partir de R en échangeant les sommets initiaux et finaux, et en inversant le sens des arcs :

$$R^{-1} := \{(X, H^{-1}) \mid (X, H) \in R\}$$

avec H^{-1} le graphe *inverse* de H :

$$\begin{aligned} H^{-1} \;&:=\; \{t \xrightarrow{a} s \mid s \xrightarrow[[H]]{a} t\} \cup (H - [H]) \\ &\cup\; \{is \mid fs \in H\} \;\cup\; \{fs \mid is \in H\}. \end{aligned}$$

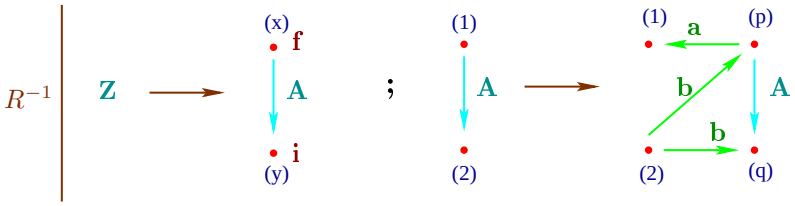


FIG. 4.14 – Inverse de la grammaire de la figure 4.6

L'inverse d'une grammaire R reconnaît donc le langage des miroirs $\tilde{u}$ des mots u reconnus par R : $L(R^{-1}) = \widetilde{L(R)}$. Et plus généralement, l'ensemble des graphes engendrés par la grammaire inverse de R est l'ensemble des graphes inverses des graphes engendrés par R : $(R^{-1})^\omega = (R^\omega)^{-1} = \{H^{-1} \mid H \in R^\omega\}$. Notons que on a $(R^{-1})^{-1} = R$.

Maintenant que nous avons défini les outils que nous allons manipuler, nous allons entrer dans le vif du sujet. En effet, nous allons nous baser sur les niveaux des sommets de graphes engendrés par des grammaires pour définir une relation de synchronisation entre grammaires. Par cette relation de synchronisation, nous allons être à même de déterminer des classes de langages algébriques qui vont s'avérer être des extensions booléennes des langages réguliers.

Chapitre 5

Synchronisation de grammaires

Dans ce chapitre, nous développons une relation de synchronisation entre grammaires de graphes. Une grammaire R synchronise une grammaire S si le graphe engendré par S est simulé niveau par niveau par R. Pour une grammaire R donnée, nous étudions les propriétés algébriques de l'ensemble des langages acceptés par les grammaires synchronisées par R. Nous prouvons que cet ensemble forme une algèbre booléenne de langages algébriques. Les travaux évoqués dans la partie contexte ci-après traitent également de la problématique d'extension des propriétés des langages réguliers en définissant des algèbres booléennes de langages algébriques déterministes. Notre concept de synchronisation requiert néanmoins que nous considérions des grammaires plus générales que les grammaires engendrant des graphes déterministes, les grammaires pondérées : deux arcs de même étiquette et de sources de même niveau ont des buts de même niveau. Mais nous montrons que les graphes pondérés sont déterminisables et nous obtenons ainsi une nouvelle manière de définir des algèbres booléennes de langages algébriques déterministes qui, sous certaines conditions, sont closes par concaténation et par étoile de Kleene.

5.1 Contexte

Il est bien connu que les langages algébriques ne sont pas clos par intersection et par complémentaire. Un exemple classique prend les langages $\{a^n b^n c^m \mid m, n \geq 0\}$ et $\{a^m b^n c^n \mid m, n \geq 0\}$ dont l'intersection est le langage $\{a^n b^n c^n \mid n \geq 0\}$ qui n'est pas algébrique. Connaissant cela, notre travail ne consiste pas à étendre les propriétés de clôture des langages réguliers aux langages algébriques, mais de trouver des sous classes (strictes) de langages algébriques pour lesquelles ces propriétés sont vérifiées. Différents travaux portant sur l'extension des langages réguliers ont été faits ces dernières années.

Les langages input-driven

En 1980, Kurt Mehlhorn considère le dallage de graphes d'automates à pile dont la hauteur de pile est fonction du temps [Mel80]. Il montre ainsi que les langages reconnus par les automates à pile déterministes *input-driven*, qui sont les automates à pile temps-réel (sans mouvement à vide) dont les transitions sont définies par la lettre d'entrée, sont reconnaissables en espace $O((\log n)^2 / \log \log n)$.

Les langages visibles

En 2004, Alur et Madhusudan [AM04] ont défini les langages visibles. Cette approche consiste à diviser l'alphabet d'entrée en trois parties : les symboles d'empilement (Σ_{push}), les symboles internes (Σ_{int}) et les symboles de dépilement (Σ_{pop}). Sont alors définis les automates à pile visibles sur cette partition, de sorte que l'ajout d'une lettre sur la pile (resp. le retrait d'une lettre, une action interne) ne peut se faire qu'à la lecture d'une lettre de Σ_{push} (resp. Σ_{pop}, Σ_{int}). Ainsi, la hauteur de pile est directement liée aux lettres présentes dans le mot lu. Les automates visibles sont donc une extension des automates input-driven. Un langage est visible si il existe un automate visible reconnaissant ce langage (en fonction d'une certaine partition).

Théorème 5.1.1. *[AM04] L'ensemble des langages visibles sur une partition*

donnée est un sous-ensemble des langages algébriques déterministes, clos par union, intersection, complémentaire, concaténation et étoile de Kleene. De plus, tout langage régulier est visible.

Synchronisation par transducteurs finis

En 2006, Caucal [Ca06] définit une relation de synchronisation d'automates à pile déterministes par un transducteur. Les transducteurs considérés sont des automates finis $M = (\mathcal{S}, \Sigma \times \mathbb{Z})$ (les entrées sont sur Σ et les sorties sur $\mathbb{Z}$) déterministes sur les entrées

$$\begin{aligned} p \xrightarrow[M]{(a,x)} q \ \wedge \ p \xrightarrow[M]{(a,y)} r \quad &\Longrightarrow \quad x = y \ \wedge \ q = r \\ iq, ir \in M \quad &\Longrightarrow \quad q = r. \end{aligned}$$

On dit aussi que M est un *transducteur séquentiel*.
Le langage reconnu par M est

$$L(M) = \{(u_1 u_2 \dots u_n, \sum_{i=1}^{n} x_i) \mid (u_1, x_1)(u_2, x_2) \dots (u_n, x_n) \in L(M, i)\}$$

dont le domaine $Dom(L(M))$ est le *langage des entrées* de M. A toute entrée $u \in Dom(L(M))$ est associé son poids $||u||$, *i.e.* $(u, ||u||) \in L(M)$. Un automate à pile déterministe A est synchronisé par un transducteur M si tout mot $u \in L(A)$ est une entrée de M et la hauteur de pile de la configuration acceptante est la valeur absolue de $||u||$. Un langage L est synchronisé par M s'il existe un automate à pile synchronisé par M et acceptant L. On définit l'ensemble $Sync(M)$ des langages synchronisés par M comme étant

$$Sync(M) = \{L(A) \mid M \ synchronise \ A\}$$

l'ensemble des langages acceptés par les automates synchronisés par M. Pour un transducteur M donné, la famille des langages synchronisés par M est une algèbre de Boole de langages algébriques déterministes.

Les automates à hauteur de pile déterministe

En 2007, Nowotka et Srba [NS07] définissent une classe d'automates à pile complets (le langage initial est l'ensemble de tous les mots) et pour

lesquels la hauteur de pile est déterminée par le mot d'entrée. En notant $N(A, w)$ l'ensemble des hauteurs de pile pour toute exécution de l'automate A sur le mot d'entrée w, un automate à hauteur de pile déterministe A vérifie $|N(A, w)| \leq 1$. Deux automates A et B sont synchronisés si pour tout mot w, $N(A, w) = N(B, w)$. Cette relation est une relation d'équivalence et l'ensemble des langages acceptés par les automates d'une même classe d'équivalence sont clos par union et intersection. De plus, si les automates sont temps-réel et déterministes, ces sous-familles de langages algébriques sont également closes par complémentaire.

Ces trois approches ont toutes un but commun : étendre des propriétés de clôture des langages réguliers à des sous-familles de langages algébriques. Bien qu'utilisant des approches différentes, les techniques d'extension ont un point commun : les langages algébriques « synchronisés » forment des sous-ensembles remarquables de langages algébriques partageant bien des propriétés de clôture avec les langages réguliers. Cette relation de « synchronisation » prend différentes formes : dans les travaux de [AM04], la synchronisation se fait entre les lettres composant un mot et la hauteur de pile de la configuration acceptante ; dans ceux de [NS07], c'est la hauteur de pile de toutes les configurations acceptantes des automates qui établit la relation ; dans les travaux de [Ca07], la synchronisation se fait entre le poids d'un mot (selon un transducteur fini donné) et la hauteur de pile de la configuration acceptante. On peut citer deux points communs à ces approches. Premièrement, la relation de synchronisation existe dans l'unicité d'un poids (la hauteur de pile) pour les mots acceptés. Le deuxième point commun est dans l'utilisation des automates à pile pour décrire des propriétés de langages algébriques. Dans le travail qui va vous être présenté ici, le premier point reste valable. Par contre, nous n'utilisons pas d'automates à pile pour établir une relation de synchronisation mais directement les grammaires de graphes.

5.2 Les grammaires pondérées

Pour un automate déterministe G engendré par une grammaire R, on définit le *poids* $||u||_R$ d'un mot initial $u \in L(G, i)$ selon R. Ce poids est donné par le niveau du dernier sommet du chemin initial étiqueté par u :

$$||u||_R := l_G^R(s) \qquad \text{tel que} \qquad \exists ir \in G,\ r \overset{u}{\underset{G}{\Longrightarrow}} s.$$

Par exemple, si on nomme R la grammaire présentée à la figure 4.6, le graphe engendré par R est présenté à la figure 4.9 où on représente également le niveau des sommets : deux sommets de même niveau sont à la verticale l'un de l'autre. Ainsi, pour tout $m \geq n \geq 0$, $||a^m b^n||_R = m - n$.

Néanmoins, nous ne nous restreignons pas aux grammaires engendrant des graphes déterministes. Mais les grammaires que nous considérons ont la propriété de définir un poids unique pour tout mot du langage initial.

Une grammaire déterministe de graphes R est une *grammaire pondérée* si les chemins initiaux de même étiquette aboutissent à des sommets de même poids :

$$s \overset{u}{\underset{G}{\Longrightarrow}} q' \ \wedge\ s' \overset{u}{\underset{G}{\Longrightarrow}} q' \ \wedge\ is, is' \in G \implies l_G^R(q) = l_G^R(q').$$

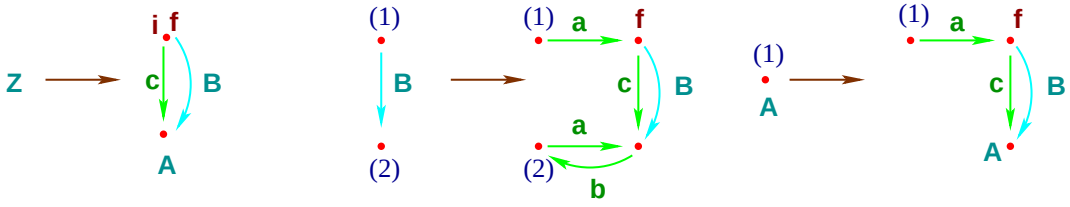


FIG. 5.1 – Une grammaire pondérée.

La grammaire figurant à la figure 5.1 nous donne un exemple de grammaire engendrant le graphe de la figure 5.2 qui est non-déterministe mais pondéré. En effet, il existe deux chemins initiaux étiquetés par le mot *caa* et le niveau de leurs buts est le même : 2.

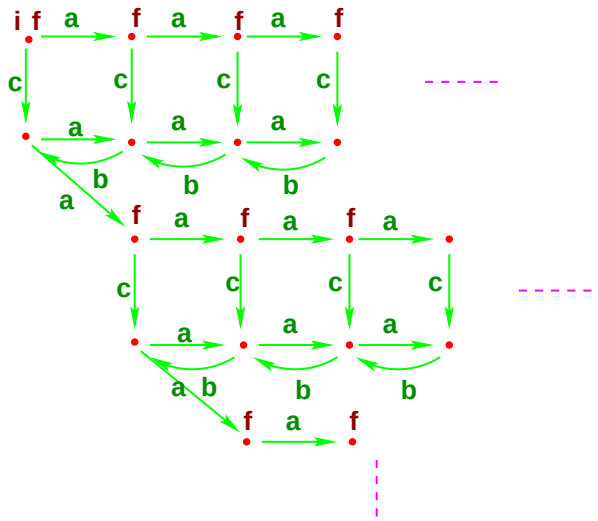


FIG. 5.2 – Graphe engendré par la grammaire pondérée de la figure 5.1.

En particulier, toute grammaire engendrant un graphe déterministe est pondérée. De plus, nous verrons au lemme 5.3.8 que toute grammaire pondérée est déterminisable.
On peut également noter que tout graphe engendré par une grammaire pondérée est de degré sortant fini (au même titre que les graphes déterministes) puisque l'alphabet des étiquettes est fini et que le nombre de sommets de même poids est aussi fini. Enfin, nous montrons que le caractère pondéré est décidable.

Lemme 5.2.1. *On peut décider si une grammaire de graphes est pondérée.*

Démonstration :
Soit R une grammaire. On suppose que chaque règle de R est de la forme $(A1 \dots \rho(A), R(A))$ pour tout $A \in N_R$. Pour tout $n \in [\rho(A)]$, on détermine l'ensemble $Out(A, n)$ des étiquettes des arcs de source n dans le graphe engendré par R à partir de l'hyperarc $A1 \dots \rho(A)$:

$$Out(A, n) \ := \ \{a \mid \exists s, \ n \xrightarrow{a} s \in R^{\omega}(A)\}.$$

Cet ensemble est le plus petit point fixe du système suivant :

$$\begin{aligned} Out(A,n) &= \{a \mid n \xrightarrow[[R(A)]]{a}\} \\ &\quad \cup \bigcup\{Out(B,m) \mid B \in N_R \ \wedge\ \exists X,\ BX \in R(A) \ \wedge\ X(m) = n\}. \end{aligned}$$

Le plus petit point fixe existe car $Out(A,n) \subseteq T_R \cap F_2$ est fini.
Pour tout $A, B \in N_R$ avec $x \in \mathcal{S}_{R(A)}$ et $y \in \mathcal{S}_{R(B)}$, on associe l'ensemble $Sync_{A,B}(x,y)$ de sommets synchronisés dans $\mathcal{S}_{R(A)} \times \mathcal{S}_{R(B)}$ à partir de (x,y) qui se définit comme le plus petit point fixe du système :

$$\begin{aligned} Sync_{A,B}(x,y) &= \{(x,y)\} \\ &\cup\ \{(t,q) \mid \exists (s,p) \in Sync_{A,B}(x,y),\ \exists a,\ s \xrightarrow[[R(A)]]{a} t \ \wedge\ p \xrightarrow[[R(B)]]{a} q\} \\ &\cup\ \{(X(m),Y(n)) \mid \exists CX \in R(A),\ \exists DY \in R(B),\ \exists j,k, \\ &\qquad (X(j),Y(k)) \in Sync_{A,B}(x,y) \\ &\qquad \wedge\ (m,n) \in Sync_{C,D}(j,k) \wedge\ m \in [\rho(C)] \ \wedge\ n \in [\rho(D)]\}. \end{aligned}$$

A partir des sommets initiaux, on calcule l'ensemble $Sync$ des sommets synchronisés. $Sync$ est le plus petit point fixe du système suivant :

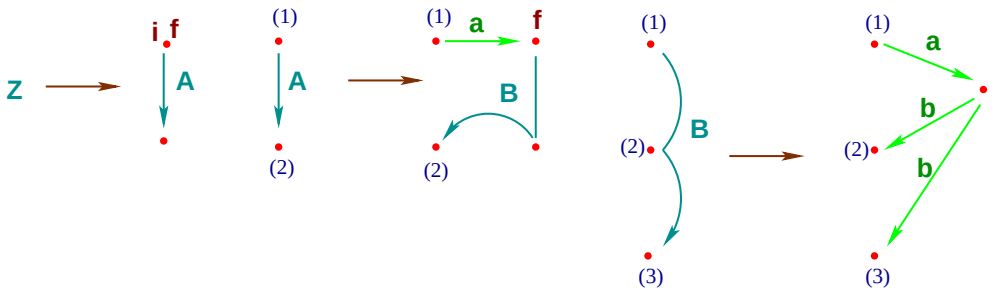
$$\begin{aligned} Sync &= \{(Z,s,Z,t) \mid is, it \in R(Z)\} \\ &\cup\ \{(A,s,B,t) \mid \exists x,y,\ (A,x,B,y) \in Sync \\ &\qquad \wedge\ (s,t) \in Sync_{A,B}(x,y)\} \\ &\cup\ \{(C,m,D,n) \mid \exists A,B \in N_R,\ \exists CX \in R(A),\ \exists DY \in R(B), \\ &\qquad (A,X(m),B,Y(n)) \in Sync\}. \end{aligned}$$

R n'est pas pondérée si et seulement si, soit on peut synchroniser un sommet d'entrée avec un sommet qui ne soit pas une entrée, soit pour $(A,s,B,t) \in Sync$, la production d'un arc de source s ne nécessite qu'une étape de récriture (par la règle pour A) alors que la production d'un arc de même étiquette de source t se fait en plusieurs étapes (par la règle pour B) :

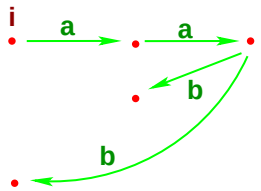
$$\begin{aligned} &\exists (A,s,B,t) \in Sync,\ (s \in [\rho(A)] \ \wedge\ t \notin [\rho(B)]) \\ \vee\ &\exists s \xrightarrow[[R(A)]]{a},\ \exists CX \in R(B),\ \exists j \in [\rho(C)],\ X(j) = t \ \wedge\ a \in Out(C,j). \end{aligned}$$

□

Illustrons la construction du lemme 5.2.1 par la grammaire R suivante :



engendrant le graphe fini ci-dessous et dont les sommets sur une même verticale sont de même niveau.



On a

$$\begin{aligned} Sync_{B,B}(1,1) &= \{(1,1),(t,t),(2,2),(2,3),(3,2),(3,3)\} \\ Sync_{A,A}(1,1) &= \{(1,1),(r,r),(s,s),(s,2),(2,s),(2,2)\} \\ Sync_{Z,Z}(p,p) &= \{(p,p),(q,q)\} \end{aligned}$$

et donc

$$\begin{aligned} Sync = \{&(Z,p,Z,p),(Z,q,Z,q),(A,1,A,1),(A,r,A,r), \\ &(A,s,A,s),(A,s,A,2),(A,2,A,s),(A,2,A,2)\} \end{aligned}$$

Comme $(A,2,A,s) \in Sync$ et $s \notin \{1,2\}$, la grammaire R n'est pas pondérée.

5.3 La synchronisation

Dans ce chapitre, après avoir défini la relation de synchronisation entre grammaires pondérées, nous décrivons les opérations de graphes qui vont nous permettre d'obtenir nos résultats. Enfin, nous montrons que les langages acceptés par les grammaires pondérées sont les langages algébriques déterministes.

Soit R une grammaire pondérée. Une grammaire pondérée S est *synchronisée* par R, et on note $R \triangleright S$ (ou symétriquement $S \triangleleft R$), si

– tout mot initial de S est un mot initial de R : $L(S,i) \subseteq L(R,i)$ et
– pour tout mot initial $u \in L(S,i)$, le poids de u défini par S est le même que celui par R : $||u||_S = ||u||_R$.

La grammaire de la figure 5.3 synchronise celle de la figure 4.6.

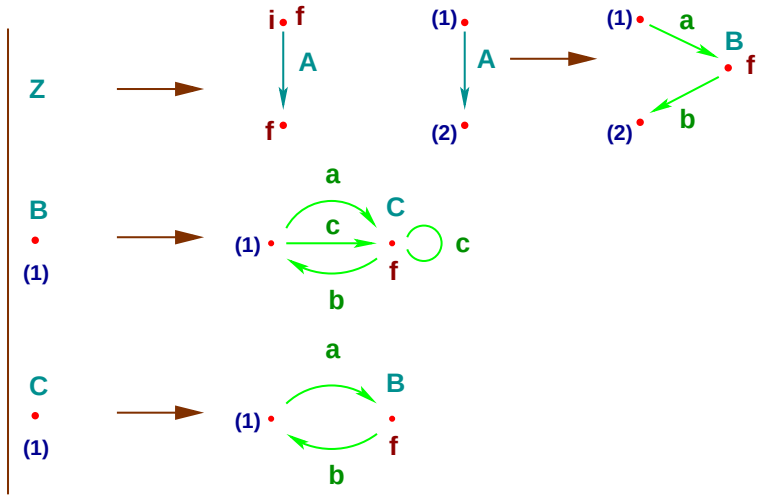


FIG. 5.3 – Un synchronisateur pour la grammaire de la figure 4.6.

Comme la relation $\triangleright$ n'est pas symétrique, on différencie R de S en appelant R un *synchronisateur*. La relation $\triangleright$ n'est pas antisymétrique mais elle est transitive et réflexive.
Deux grammaires R et S sont *bi-synchronisées* et on note $S \bowtie R$ si S et R sont des synchronisateurs l'un pour l'autre. Par ailleurs, elles reconnaissent le même langage initial :

$$R \bowtie S \iff R \triangleright S \wedge L(R,i) = L(S,i).$$

Une première propriété de la relation $\triangleright$ concerne le degré des graphes. En effet, $\triangleright$ préserve la finitude des degrés des graphes engendrés.

Lemme 5.3.1. *$R \triangleright S \wedge R^\omega$ de degré fini $\wedge$ S^ω accessible (à partir de i) $\Longrightarrow$ S^ω de degré fini.*

Démonstration :
Nous prouvons cette propriété par contraposée : supposons $R \triangleright S$ et S^ω de degré infini. Montrons que R^ω est de degré infini.
Soient G et H des graphes engendrés respectivement par R et S. Par le lemme 5.3.8, on peut considérer G et H déterministes. Il existe donc un sommet $p \in \mathcal{S}_H$ de degré entrant infini. De plus, H étant régulier, le nombre de sommets de même niveau est fini :

$$\forall n \geq 0,\ |\{s \in \mathcal{S}_H \mid l_H^S(s) = n\}| < \omega\ .$$

Donc il existe une infinité de sommets $\{p_0, \ldots\} \subseteq S_H$ de poids différents

$$l_H^S(p_0) < l_H^S(p_1) < \ldots \text{ tel que } p_k \xrightarrow[H]{a_k} p$$

pour tout $k \geq 0$ avec $a_k \in E_H$.
Comme S^ω est accessible à partir de i, on prend pour tout $n \geq 0$ un mot $u_n \in L(S, i)$ étiquetant un chemin initial d'arrivée p_n :

$$\exists ir \in H,\ \forall n \geq 0,\ r \xRightarrow[H]{u_n} p_n.$$

Comme R synchronise S, il existe dans G un ensemble $\{q_0, q_1, \ldots\}$ tel que :

$$\exists ir' \in G,\ \forall k \geq 0,\ r' \xRightarrow[G]{u_k} q_k \xrightarrow[G]{a_k} s_k\ \wedge\ l_G^R(q_k) = l_H^S(p_k)\ \wedge\ l_G^R(s_k) = l_H^S(p).$$

Comme le nombre de sommets de G ayant le même poids est fini

$$|\{s_n \in \mathcal{S}_G \mid l_G^R(s_n) = l_H^S(p)\}| < \omega,$$

il existe un sommet $s \in \mathcal{S}_G$ et une suite infinie $j_0 < j_1 < \ldots$ tels que pour tout $k \geq 0$, $q_{j_k} \xrightarrow[G]{a_{j_k}} s$. Comme les niveaux des sommets q_{j_k} sont tous différents, s est de degré entrant infini.

□

Au cours de notre étude, il s'est avéré plus commode de ne considérer que des automates réguliers dont les sommets initiaux sont de niveau 0. La propriété 5.3.2 décrit comment on transforme une grammaire et son synchronisateur en des grammaires synchronisées équivalentes dont les sommets initiaux sont au niveau 0, quand le nombre de sommets initiaux est fini.

Propriété 5.3.2. *On peut transformer toute grammaire R (où $|\mathcal{S}_{R^\omega,i}| < \omega$ en une grammaire équivalente R' de sommet initial dans l'axiome et telle que*

$$\{S^\omega \mid S \lhd R\} \;=\; \{S'^\omega \mid S' \lhd R'\}.$$

Démonstration :
Soit $Z \to G_0$ la règle axiome d'une grammaire R. Soit $G_0 \underset{R}{\Rightarrow} \ldots \underset{R}{\Rightarrow} G_n \underset{R}{\Rightarrow} \ldots$ une dérivation parallèle infinie à partir de G_0. Soit P l'ensemble des niveaux des sommets de $\bigcup_n [G_n]$ coloriés par i. On considère la grammaire

$$R' \;:=\; (R - \{(Z, G_0)\}) \cup \{(Z, G_{max\{n \in \mathbf{N} \mid n \in P\})}\}.$$

Ainsi $R'^\omega = R^\omega$ et les sommets initiaux de R' sont dans son axiome. De plus, pour $S \lhd R$, on a $S' \lhd R'$ et $S^\omega = S'^\omega$.

□

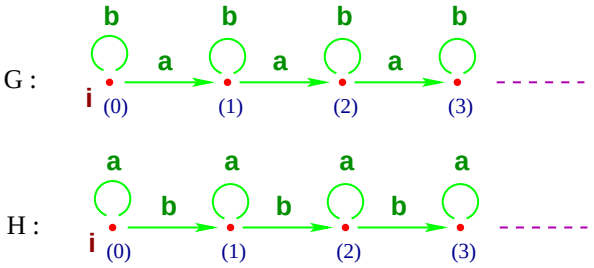
Une opération usuelle sur les graphes finis (notamment utilisée pour montrer la clôture par intersection des langages réguliers) est le produit de synchronisation de graphes. On donne ici une extension de ce produit et on y rajoute un coloriage pour préserver l'information des sommets finaux des graphes dont on fait le produit.
Le produit de synchronisation $G \times H$ des graphes G et H est défini par

$$\begin{aligned} G \times H \;:=\; & \{(s,p) \xrightarrow{a} (t,q) \mid s \underset{G}{\xrightarrow{a}} t \;\wedge\; p \underset{H}{\xrightarrow{a}} q\} \\ \cup\; & \{i(s,p) \mid is \in G \wedge ip \in H\} \cup \{f(s,p) \mid fs \in G \wedge fp \in H\} \\ \cup\; & \{f_1(s,p) \mid fs \in G \wedge p \in \mathcal{S}_H \wedge fp \notin H\} \\ \cup\; & \{f_2(s,p) \mid s \in \mathcal{S}_G \wedge fs \notin G \wedge fp \in H\} \end{aligned}$$

où f_1 et f_2 sont deux nouvelles couleurs finales.
Le produit de synchronisation ne préserve pas la régularité des graphes. Il suffit de considérer les graphes réguliers suivants :

$$\begin{aligned} G \;&=\; \{n \xrightarrow{a} n+1 \mid n \in \mathbb{N}\} \cup \{n \xrightarrow{b} n \mid n \in \mathbb{N}\} \cup \{i0\} \\ et\quad H \;&=\; \{n \xrightarrow{b} n+1 \mid n \in \mathbb{N}\} \cup \{n \xrightarrow{a} n \mid n \in \mathbb{N}\} \cup \{i0\} \end{aligned}$$

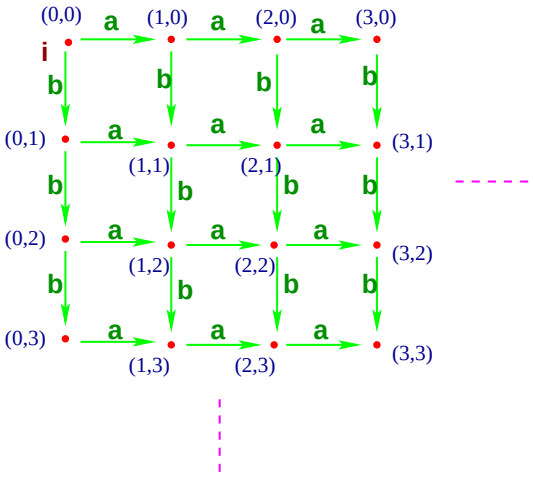
représentés ci-après :



Le produit de synchronisation $G \times H$ est le graphe non régulier suivant :

$$\begin{aligned} G \times H &= \{(m,n) \xrightarrow{a} (m+1,n) \mid m,n \in \mathbb{N}\} \\ &\cup \{(m,n) \xrightarrow{b} (m,n+1) \mid m,n \in \mathbb{N}\} \\ &\cup \{i(0,0)\} \end{aligned}$$

que l'on représente ci-dessous :



Nous restreignons ce produit de synchronisation pour ne travailler que niveau par niveau. Etant données des grammaires R et S, on définit le *produit de synchronisation par niveau* de leurs graphes engendrés par

$$R^\omega \times_l S^\omega \;:=\; \{K \mid \exists G \in R^\omega, \exists H \in S^\omega,\; K \simeq G \times_l H\}$$

où le *produit de synchronisation par niveau* $G \times_l H$ de $G \in R^\omega$ et $H \in S^\omega$ est

$$G \times_l H \;:=\; (G \times H)_{|\{(s,p) \;\mid\; l_G(s)=l_H(p)\}}.$$

Construisons une grammaire $R \times_l S$ engendrant $R^\omega \times_l S^\omega$. A tout couple $(A, B) \in N_R \times N_S$ de non terminaux de chaque grammaire et à toute relation $E \subseteq [\rho(A)] \times [\rho(B)]$ sur les entrées dites de même niveau :

$$E(i) \cap E(j) \neq \emptyset \implies E(i) = E(j) \qquad \text{pour tout } i, j \in [\rho(A)],$$

on associe un nouveau symbole $[A, B, E]$ d'arité $|E|$, sauf que $[Z, Z, \emptyset] = Z$. A chaque hyperarc non-terminal $Ar_1 \dots r_m$ de R ($A \in N_R$ et $m = \rho(A)$) et à chaque hyperarc non-terminal $Bs_1 \dots s_n$ de S ($B \in N_S$ et $n = \rho(B)$), on associe l'hyperarc suivant :

$$\begin{aligned}[Ar_1 \dots r_m, Bs_1 \dots s_n, E] \;=\; & [A, B, E](r_1, s_1)_E \dots (r_1, s_n)_E \dots \\ & (r_m, s_1)_E \dots (r_m, s_n)_E\end{aligned}$$

$$\text{avec } (r_i, s_j)_E := \begin{cases} (r_i, s_j) & \text{si } (i, j) \in E \\ \varepsilon & \text{sinon.} \end{cases}$$

Le *produit de synchronisation par niveau* $R \times_l S$ de R par S est la grammaire constituée des règles ci-dessous :

$$\begin{aligned}[AX, BY, E] \;\to\; & ([P] \times [Q])_{|\overline{E}} \\ \cup\; & \{[CU, DV, E'] \mid CU \in P \wedge C \in N_R \wedge DV \in Q \wedge D \in N_S\}\end{aligned}$$

pour toutes règles $(AX, P) \in R$ et $(BY, Q) \in S$, et pour tout $E \subseteq [\rho(A)] \times [\rho(B)]$ avec

$$\begin{aligned}\overline{E} \;&:=\; \{(X(i), Y(j)) \mid (i, j) \in E\} \cup (\mathcal{S}_P - \mathcal{S}_X) \times (\mathcal{S}_Q - \mathcal{S}_Y) \\ \text{et}\quad E' \;&:=\; \{(i, j) \in [\rho(C)] \times [\rho(D)] \mid (U(i), V(j)) \in \overline{E}\}.\end{aligned}$$

En restreignant $R \times_l S$ aux non-terminaux accessibles à partir de Z, on obtient une grammaire engendrant le produit de synchronisation par niveau de leurs graphes engendrés.
Pour illustrer cette construction, nous prenons deux grammaires S et T données à la figure 5.4.

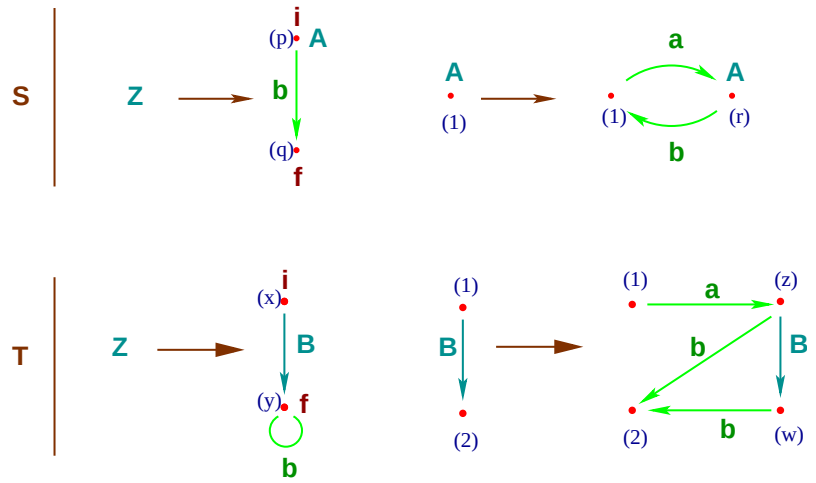


FIG. 5.4 – Deux grammaires synchronisées par un même synchronisateur.

Le produit de synchronisation par niveau $S \times_l T$ est présenté à la figure 5.5 où $C = [A, B, \{\{1,1\},\{1,2\}\}]$.

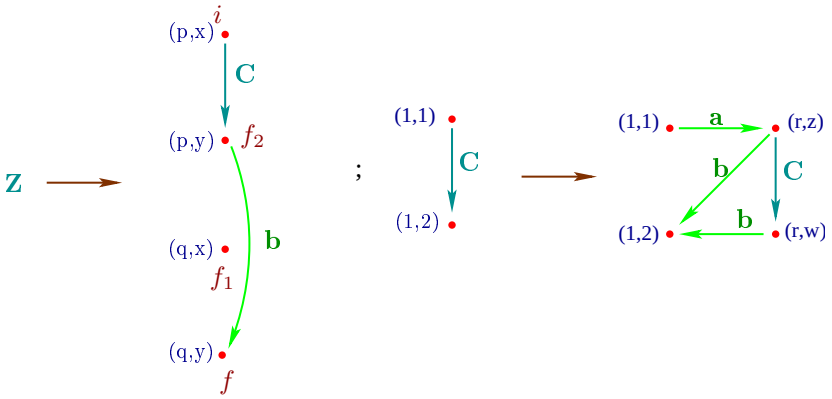


FIG. 5.5 – Le produit de synchronisation par niveau $S \times_l T$.

Cette grammaire $R \times_l S$ engendre le produit de synchronisation par niveau $R^\omega \times_l S^\omega$ des graphes engendrés par R avec ceux engendrés par S.

Lemme 5.3.3. *Pour toutes grammaires R et S, $(R \times_l S)^\omega = R^\omega \times_l S^\omega$.*

Démonstration :

Soient $G \in R^\omega$ et $H \in S^\omega$. Il existe une dérivation $Z = G_0 \underset{R}{\Rightarrow} G_1 \ldots \underset{R}{\Rightarrow} G_n \ldots$

telle que $\bigcup_{n\geq 0}[G_n] = G$ et une dérivation $Z = H_0 \underset{S}{\Rightarrow} H_1 \ldots \underset{S}{\Rightarrow} H_n \ldots$ telle que $\bigcup_{n\geq 0}[H_n] = H$.

i) Vérifions que $\bigcup_n [G_n] \times [H_n] = G \times H$.

Pour tout $n \geq 0$, $[G_n] \subseteq G$ et $[H_n] \subseteq H$.

Donc $[G_n] \times [H_n] \subseteq G \times H$ d'où $\bigcup_n [G_n] \times [H_n] \subseteq G \times H$.

Réciproquement, soit $(s,p) \underset{G\times H}{\overset{a}{\rightarrow}} (t,q)$. On a donc $s \underset{G}{\overset{a}{\rightarrow}} t$ et $p \underset{H}{\overset{a}{\rightarrow}} q$.

Soit k (resp. l) le plus petit entier tel que $s \xrightarrow{a} t \in G_k$ (resp. $p \xrightarrow{a} q \in H_l$).

Pour $m := max\{k,l\}$, on a $s \underset{G_m}{\overset{a}{\rightarrow}} t$ et $p \underset{H_m}{\overset{a}{\rightarrow}} q$, donc

$$(s,p) \xrightarrow{a} (t,q) \in [G_m] \times [H_m].$$

De façon identique pour tout $c(s,p) \in G \times H$, il existe m tel que

$$c(s,p) \in [G_m] \times [H_m].$$

ii) On veut montrer que $G \times_l H \in (R \times_l S)^\omega$.

Posons $E = \{(s,p) \in \mathcal{S}_G \times \mathcal{S}_H \mid l_G(s) = l_H(p)\}$. On a

$$\begin{aligned} G \times_l H &= (G \times H)_{|E} \\ &= (\bigcup_{n\geq 0} [G_n] \times [H_n])_{|E} \quad \text{d'après (i)} \\ &= \bigcup_{n\geq 0} ([G_n] \times [H_n])_{|E}. \end{aligned}$$

On étend le produit de synchronisation par niveau des graphes aux hypergraphes.

Soient un hypergraphe P d'ensemble d'étiquettes $F_P \subseteq N_P \cup F_1 \cup F_2$ et une application $l_P : \mathcal{S}_P \rightarrow \mathbb{N}$.

Soient un hypergraphe Q d'ensemble d'étiquettes $F_Q \subseteq N_Q \cup F_1 \cup F_2$ et une application $l_Q : \mathcal{S}_Q \rightarrow \mathbb{N}$.

On définit le produit de synchronisation par niveau $P \times_l Q$ de P avec Q comme suit :

$$\begin{aligned} P \times_l Q \;:=\; & ([P] \times [Q])_{|\{(p,q) \mid l_G(p) = l_H(q)\}} \\ \cup\; & \{[A,B,E](r_1,s_1)_E \ldots (r_m,s_n)_E \mid A \in N_R \wedge B \in N_S \\ & \qquad \wedge\, Ar_1 \ldots r_m \in P \wedge Bs_1 \ldots s_n \in Q\} \end{aligned}$$

avec $E = \{(i,j) \in [\rho(A)] \times [\rho(B)] \mid l_P(r_i) = l_Q(s_j)\}$.
En particulier $[P \times_l Q] = [P] \times_l [Q]$.
Soient $P \underset{R}{\Rightarrow} P'$ et $Q \underset{S}{\Rightarrow} Q'$ et posons

$$m \ := \ max(\{l_P(s) \mid s \in \mathcal{S}_P\} \cup \{l_Q(s) \mid s \in \mathcal{S}_Q\}).$$

On définit

$$l_{P'}(s) := \begin{cases} l_P(s) & \text{si} \quad s \in \mathcal{S}_P \\ m+1 & \text{si} \quad s \in \mathcal{S}_{P'} - \mathcal{S}_P \end{cases}$$

et

$$l_{Q'}(s) := \begin{cases} l_Q(s) & \text{si} \quad s \in \mathcal{S}_Q \\ m+1 & \text{si} \quad s \in \mathcal{S}_{Q'} - \mathcal{S}_Q. \end{cases}$$

Ainsi, $P \times_l Q \underset{R \times_l S}{\Longrightarrow} P' \times_l Q'$.
Par récurrence sur $n \geq 0$, on obtient $G_n \times_l H_n \underset{R \times_l S}{\Longrightarrow} G_{n+1} \times_l H_{n+1}$.
Ainsi, $\bigcup\limits_{n \geq 0} [G_n \times_l H_n] \in (R \times_l S)^\omega$. Finalement,

$$G \times_l H = \bigcup_{n \geq 0} ([G_n] \times [H_n])_{|E} = \bigcup_{n \geq 0} [G_n \times_l H_n] \in (R \times_l S)^\omega.$$

□

Le produit de synchronisation par niveau des automates réguliers est une généralisation du produit de synchronisation standard des automates finis : en considérant que tous les sommets d'un automate fini ont le même niveau (par exemple 0) les deux produits effectuent la même opération.

Le produit de synchronisation $G \times_l H$ des graphes G et H ne conserve que les arcs de G et de H qui peuvent être synchonisés. Donnons un produit plus général permettant de conserver tous les arcs de G et H.
On prend deux nouvelles couleurs, f^1 et f^2, et un nouveau symbole $\perp$.
Le *produit de synchronisation généralisé* $G \otimes H$ de graphes G et H est défini par

$$\begin{array}{rcl} G \otimes H & := & G \times H \\ & \cup & \{(s,p) \xrightarrow{a} (t,\bot) \mid s \xrightarrow[G]{a} t \wedge p \in \mathcal{S}_H \cup \{\bot\} \wedge \neg(\exists q,\ p \xrightarrow[H]{a} q)\} \\ & \cup & \{(s,p) \xrightarrow{a} (\bot,q) \mid p \xrightarrow[H]{a} q \wedge s \in \mathcal{S}_G \cup \{\bot\} \wedge \neg(\exists t,\ s \xrightarrow[G]{a} t)\} \\ & \cup & \{f^1(s,\bot) \mid fs \in G\} \cup \{f^2(\bot,p) \mid fp \in H\} \end{array}$$

Comme pour le produit de synchronisation standard, on restreint ce nouveau produit aux sommets de même niveau. Etant données des grammaires R et S, on définit le produit de synchronisation généralisé par niveau de leurs graphes engendrés par

$$R^\omega \otimes_l S^\omega \ := \ \{K \mid \exists G \in R^\omega, \exists H \in S^\omega,\ K \simeq G \otimes_l H\}$$

où le *produit de synchronisation généralisé par niveau* $G \otimes_l H$ de $G \in R^\omega$ et $H \in S^\omega$ est

$$\begin{array}{rcl} G \otimes_l H & := & G \times_l H \\ & \cup & \{(s,p) \xrightarrow{a} (t,\bot) \mid s \xrightarrow[G]{a} t \wedge ((s,p) \in \mathcal{S}_{G\times_l H} \vee p = \bot) \\ & & \qquad \wedge \forall q(p \xrightarrow[H]{a} q \implies l_G(t) \neq l_H(q))\} \\ & \cup & \{(s,p) \xrightarrow{a} (\bot,q) \mid p \xrightarrow[H]{a} q \wedge ((s,q) \in \mathcal{S}_{G\times_l H} \vee s = \bot) \\ & & \qquad \wedge \forall t(s \xrightarrow[G]{a} t \implies l_G(t) \neq l_H(q))\} \\ & \cup & \{f^1(s,\bot) \mid fs \in G\} \cup \{f^2(\bot,p) \mid fp \in H\}. \end{array}$$

La définition précédente, $R \times_l S$, est étendue pour définir une grammaire $R \otimes_l S$.

On considère que tout les sommets des membres droits des règles sont tous des sommets de sortie. Pour cela, il suffit de prendre un nouveau terminal $T \in N_1$ d'arité 1 et on prend les grammaires R' et S' telles que

$$\begin{array}{rcl} R' & := & \{(X, H \cup \{Ts \mid s \in \mathcal{S}_H - \mathcal{S}_{H-[H]}\}) \mid (X,H) \in R\} \\ & \cup & \{(T1, \{T1\})\}. \end{array}$$

Cette normalisation est nécessaire pour que la récriture de tout non-terminal d'une des grammaires puisse être parallélisé par un non-terminal de l'autre. Notons ainsi que pour le cas où R et S sont synchronisés par un même synchronisateur, cette complétion n'est pas nécessaire.

Reprenant la définition de $R \times_l S$, on considère maintenant que le symbole $[A,B,E]$ est d'arité $|E| + \rho(A) + \rho(B)$ avec la définition

$$[Ar_1 \dots r_m, Bs_1 \dots s_n, E] \;:=\; [A, B, E](r_1, s_1)_E \dots (r_m, s_n)_E \\ (r_1, \bot) \dots (r_m, \bot)(\bot, s_1) \dots (\bot, s_n)$$

et dans le membre droit de la règle de $[AX, BY, E]$, on remplace

$$([P] \times [Q])_{|\overline{E}} \qquad \text{par} \qquad ([P] \otimes [Q])_{|\overline{E}} \cup \mathcal{S}_P \times \{\bot\} \cup \{\bot\} \times \mathcal{S}_Q .$$

Pour les grammaires S et T de la figure 5.4, le produit de synchronisation généralisé par niveau $S \otimes_l T$ est représenté à la figure 5.6 où
$C = [A, B, \{1, 1\}, \{1, 2\}]$.

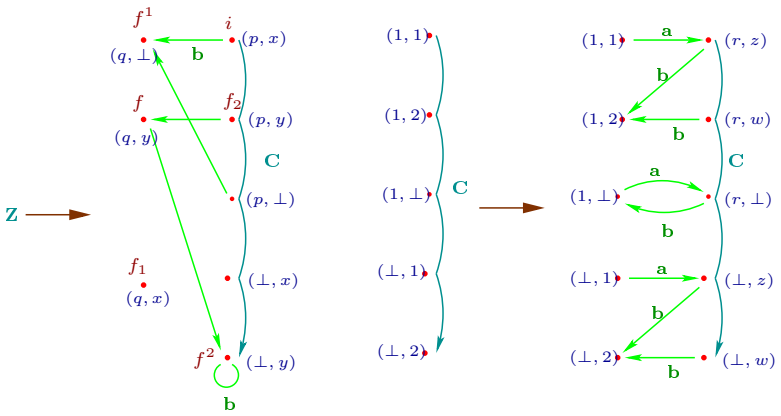
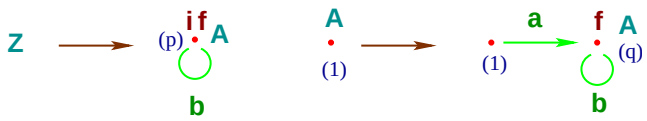
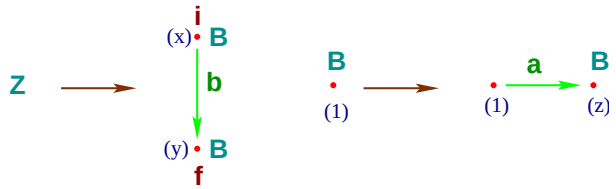


FIG. 5.6 – Le produit de synchronisation généralisé par niveau $S \otimes_l T$.

Le lemme 5.3.3 n'est plus vrai pour ce nouveau produit : en général on a $(R \otimes_l S)^\omega \neq R^\omega \otimes_l S^\omega$.
Par exemple, prenons la grammaire R suivante :



et la grammaire $S \triangleleft R$ suivante :



Le graphe $(R^\omega \otimes_l S^\omega)_{i\to}$ obtenu par application de $\otimes_l$ aux graphes engendrés par R et S et le graphe $((R\otimes_l)^\omega S)_{i\to}$ engendré par la grammaire $R \otimes_l S$ sont représentés à la figure 5.7, en se restreignant aux accessibles par i dans les deux cas.

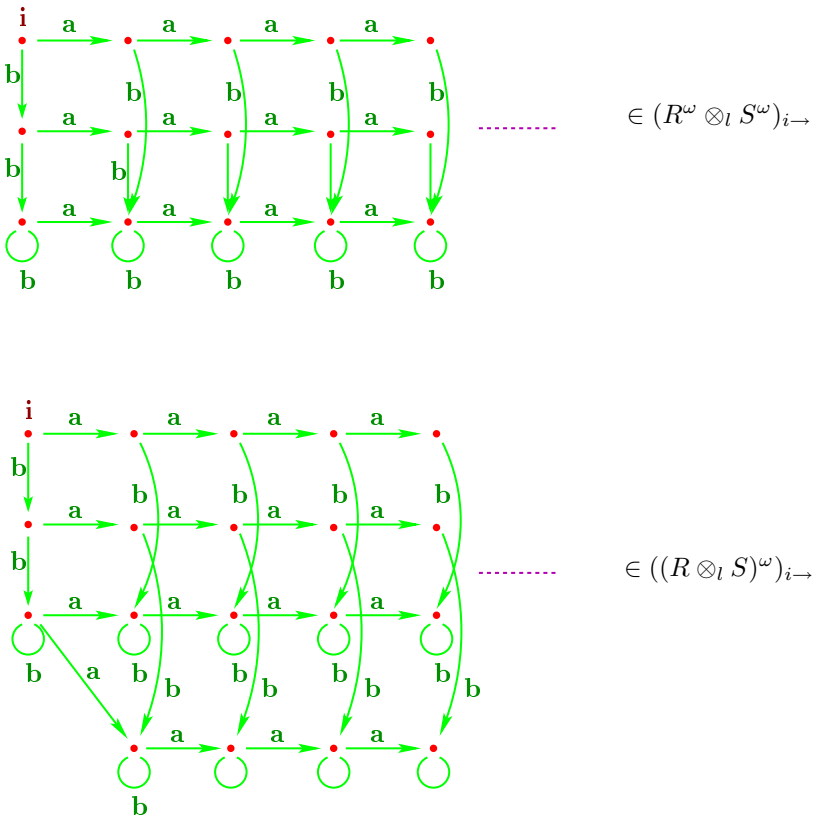


FIG. 5.7 – $(R \otimes_l S)^\omega \neq R^\omega \otimes_l S^\omega$

Pour tout $C \subseteq F_1 - \{i\}$ et toute grammaire R, on note R_C la grammaire obtenue à partir de R en remplaçant la couleur des sommets coloriés dans C par f et en enlevant f sur les autres sommets :

$$R_C := \{(X, (H - \{f\}\mathcal{S}_H) \cup \{fp \mid \exists c \in C,\ cp \in H\}) \mid (X, H) \in R\}.$$

Bien que $(R\otimes_l S)^\omega$ et $R^\omega \otimes_l S^\omega$ sont différents, ils sont bisimilaires par niveau.

Lemme 5.3.4. *Pour toutes grammaires R et S synchronisées par un même synchronisateur, on a*

$$((R \otimes_l S)_{\{f,f_1,f^1\}})_{i\to f} \bowtie R\,;\ ((R \otimes_l S)_{\{f,f_2,f^2\}})_{i\to f} \bowtie S\,;$$
$$((R \otimes_l S)_{\{c\}})_{i\to f} \bowtie ((R \times_l S)_{\{c\}})_{i\to f}\ \forall c \in \{f, f_1, f_2\}$$

Le produit de synchronisation étendu permet de montrer que l'on peut décider si une grammaire est synchronisée par une autre.

Lemme 5.3.5. *La relation $\triangleright$ est récursive.*

La démonstration de ce lemme est donnée dans un cadre plus général par celle du lemme 6.3.9.

La caractérisation de la synchronisation en terme de graphes nous amène à voir cette relation de la manière suivante : si R est un synchronisateur pour S alors, grossièrement parlant, les chemins initiaux du graphe H engendré par S « grandissent »à la même vitesse que dans le graphe G engendré par R. Ceci est illustré à la figure 5.8 dans laquelle sont représentés le graphe engendré par la grammaire de la figure 4.6 ainsi que le graphe engendré par le synchronisateur de la figure 5.3.

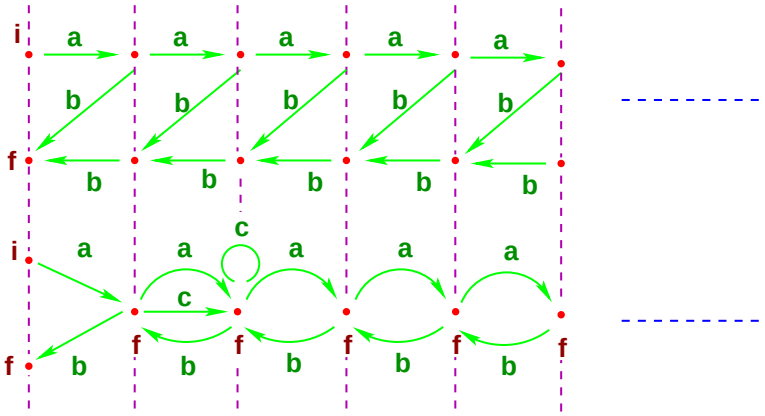


FIG. 5.8 – Le graphe synchronisé croît comme celui du synchronisateur.

Remarque : On retrouve cette notion de « croissance synchronisée »sur des piles dans [NS07] par définition des automates à hauteur de pile déterministe, dans [AM04] comme conséquence de la partition de l'alphabet et dans [Ca06] par identification entre le poids d'un mot et la hauteur de pile.

La propriété 5.3.6 illustre cette notion. En effet, si le graphe engendré par un synchronisateur R est fini, alors le graphe de toute grammaire synchronisée par R est fini.

Propriété 5.3.6. *Pour toute grammaire R engendrant un graphe fini, si $R \triangleright S$ alors S engendre un graphe fini.*

Démonstration :
Soit R une grammaire engendrant un graphe G fini et soit une grammaire S engendrant un graphe H telle que $R \triangleright S$.
Pour tout $p \in \mathcal{S}_H$, il existe un sommet $s \in \mathcal{S}_G$ tel que $l_H^S(p) = l_G^R(s)$. Comme le nombre de sommets de même poids est fini, le graphe H est fini.

□

Comme nous l'avons dit précédemment, les grammaires pondérées sont déterminisables : pour chaque grammaire pondérée R, il existe une grammaire R_{det} bisynchronisée avec R (en particulier, $L(R_{det}) = L(R)$) telle que le graphe engendré par R_{det} soit déterministe. Nous avons vu une procédure de déterminisation pour les automates finis mais sa mise en œuvre sur les automates réguliers ne nous permet pas forcément d'obtenir des automates réguliers : la classe des automates réguliers n'est pas close par l'opération de déterminisation Det qui à un graphe G associe le graphe déterministe suivant :

$$\begin{aligned} Det(G) \;:=\; & (\{P \xrightarrow{a} \{q \mid \exists p \in P,\ p \xrightarrow{a} q\} \mid P \subseteq S_G \ \wedge\ \exists p \in P,\ p \xrightarrow{a}\} \\ & \cup\ \{i\{p \mid ip \in G\}\} \ \cup\ \{fP \mid \exists p \in P,\ fp \in G\})_{i\rightarrow} \end{aligned}$$

et qui est restreint par accessibilité à partir de i.
Cette construction appliquée au graphe régulier G de la figure 5.2 donne le graphe $Det(G)$ non régulier présenté à la figure 5.9.

Pour déterminiser les grammaires pondérées, nous utilisons les grammaires dites de chemins. Une *grammaire de chemins* R est une grammaire d'arcs $F_R \subseteq F_1 \cup F_2 \cup \{Z\}$ telle que pour toute règle $(A12, R_A)$, la première (respectivement la deuxième) entrée de R_A n'est but (respectivement source) d'aucun arc terminal :

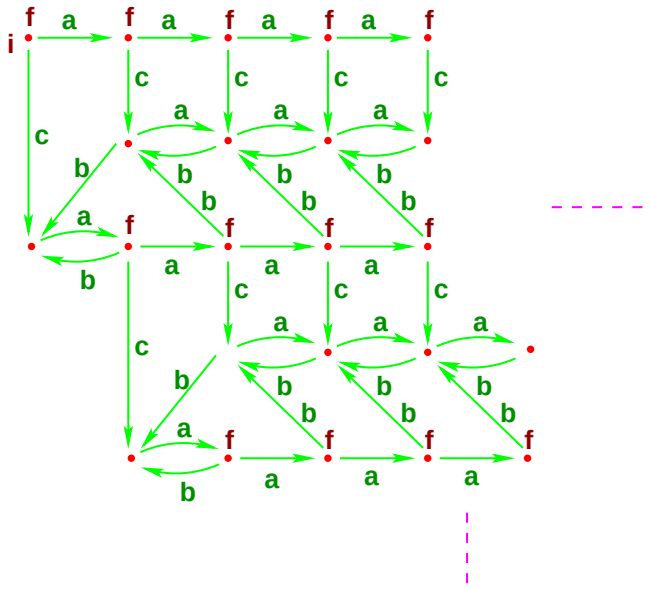


FIG. 5.9 – L'opération *Det* appliquée au graphe de la figure 5.2.

$$\forall (A12, R_A) \in R,\ \forall a \in T_R \cap F_2,\ \overset{a}{\underset{[R_A]}{\nrightarrow}} 1 \ \wedge\ 2 \overset{a}{\underset{[R_A]}{\nrightarrow}}$$

et pour toute règle $(A1, R_A)$, l'entrée de R_A n'est but d'aucun arc terminal :

$$\forall (A1, R_A) \in R,\ \forall a \in T_R \cap F_2,\ \overset{a}{\underset{[R_A]}{\nrightarrow}} 1.$$

La déterminisation

La déterminisation d'une grammaire pondérée R s'effectue en trois étapes. On commence par normaliser R en une grammaire d'arcs : tout hyperarc non-terminal est scindé en arcs non-terminaux. Puis, on transforme la grammaire normalisée en une *grammaire de chemins* pondérée $\prec R \succ$. Enfin, la grammaire $\prec R \succ$ est transformée en une grammaire R_{det} engendrant un graphe déterministe telle que $R_{det} \bowtie R$ et $L(\prec R \succ) = L(R_{det})$. Dans le lemme suivant sont regroupés les deux premières étapes de la construction.

Lemme 5.3.7. *Pour toute grammaire R, il existe une grammaire de chemins $\prec R \succ$ telle que $R \bowtie \prec R \succ$ et $L(\prec R \succ) = L(R)$.*

Démonstration :

Soit R une grammaire pondérée. On considère que les règles de R sont de la

forme $A1 \dots \rho(A) \to R_A$ et on suppose que 0 n'est pas un sommet de R. On prend deux nouveaux ensembles de fonctions non-terminales : un ensemble $N_1 := \{A_i \mid A \in N_R \wedge 1 \leq i \leq \rho(A)\} \subseteq F_1 - F_R$ de symboles d'arité 1 et un ensemble $N_2 := \{A_{i,j} \mid A \in N_R \wedge 1 \leq i, j \leq \rho(A)\} \subseteq F_2 - F_R$ de symboles d'arité 2. Le *découpage* $\prec G \succ$ d'un hypergraphe G est le graphe défini par :

$$\begin{array}{rcl}
\prec G \succ & := & [G] \\
& \cup & \{s \overset{A_{i,j}}{\to} t \mid A \in N_R \ \wedge \ 1 \leq i, j \leq \rho(A) \ \wedge \ \exists s_1, \dots, s_{\rho(A)}, \\
& & \qquad As_1 \dots s_{\rho(A)} \in G \wedge s_i = s \wedge s_j = t\} \\
& \cup & \{A_i s \mid A \in N_R \ \wedge \ 1 \leq i \leq \rho(A) \ \wedge \ \exists s_1, \dots, s_{\rho(A)}, \\
& & \qquad As_1 \dots s_{\rho(A)} \in G \wedge s_i = s\}.
\end{array}$$

On nomme $0 \notin S_G$ un nouveau sommet. Pour tout sommet $p, q \in S_G$ et toute partie $P \subseteq S_G$, on définit les graphes suivants :

$$\begin{array}{rcll}
G_{p,q,P} & := & (\{s \underset{\prec G \succ}{\overset{a}{\to}} t \mid a \in F_2 \ \wedge \ t \neq p \ \wedge \ s \neq q \ \wedge \ s, t \notin P\} & \\
& & \quad \cup (\prec G \succ \cap (F_1 - N_1)\mathcal{S}_G))_{|\{s \ \mid \ p \to^* s \to^* q\}} & \text{pour } p \neq q \\
G_{p,p,P} & := & (\{s \underset{\prec G \succ}{\overset{a}{\to}} t \mid a \in F_2 \ \wedge \ t \neq p \ \wedge \ s, t \notin P\} \ \cup \ \{s \overset{a}{\to} 0 \mid s \underset{\prec G \succ}{\overset{a}{\to}} p\} & \\
& & \quad \cup (\prec G \succ \cap (F_1 - N_1)\mathcal{S}_G))_{|\{s \ \mid \ p \to^* s \to^* 0\}} & \\
G_{p,P} & := & (\{s \underset{\prec G \succ}{\overset{a}{\to}} t \mid a \in F_2 \ \wedge \ t \neq p \ \wedge \ s, t \notin P\} & \\
& & \quad \cup (\prec G \succ \cap F_1 S_G))_{|\{s \ \mid \ p \to^* s \ \wedge \ \forall q \in (P \ \cup \ \{p\}), \ \neg(s \to^* q)\}}. &
\end{array}$$

Le découpage $\prec R \succ$ de la grammaire R est la grammaire de chemins pondérée donnée par :

$$\begin{array}{rcl}
\prec R \succ & := & \{Z \to \prec R_Z \succ\} \\
& \cup & \{A_{i,j}12 \to h_{i,j}((R_A)_{i,j,[\rho(A)]-\{i,j\}}) \mid A \in N_R \ \wedge 1 \leq i, j \leq \rho(A)\} \\
& \cup & \{A_i 1 \to h_{i,i}((R_A)_{i,[\rho(A)]-\{i\}}) \mid A \in N_R \ \wedge 1 \leq i \leq \rho(A)\}
\end{array}$$

avec $h_{i,j}$ un renommage de sommets défini par

$$\begin{array}{l}
h_{i,j}(i) = 1, \ h_{i,j}(j) = 2, \ h_{i,j}(x) = x \text{ sinon}, \quad \text{avec } i \neq j \\
h_{i,i}(i) = 1, \ h_{i,i}(0) = 2, \ h_{i,i}(x) = x \text{ sinon.}
\end{array}$$

Ainsi, R et $\prec R \succ$ sont bi-synchronisés et reconnaissent le même langage :

$$R \bowtie \prec R \succ \text{ et } L(R, i, f) \;=\; L(\prec R \succ, i, f).$$

Finalement, on peut mettre $\prec R \succ$ sous une forme réduite en supprimant tout non-terminal $A_{i,j}$ engendrant un graphe sans chemin de i à j, et tout non-terminal A_i engendrant un graphe sans état final.

□

Cette construction appliquée à la grammaire pondérée de la figure 5.1 produit la grammaire de chemin présentée à la figure 5.10.

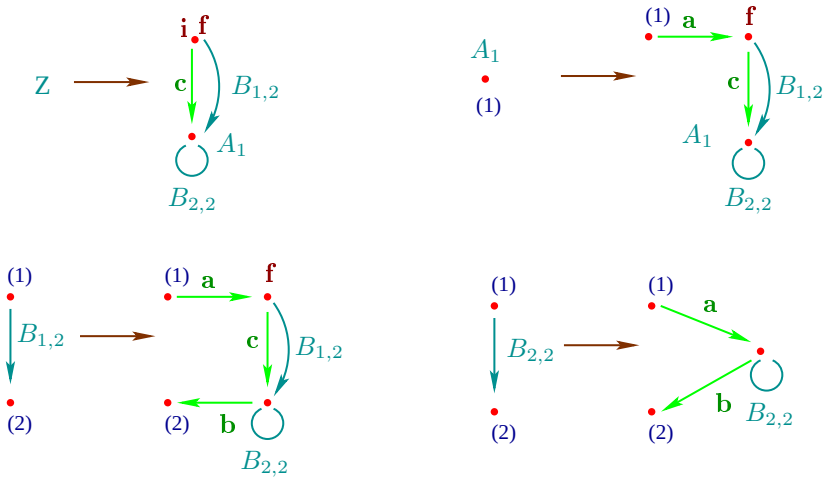


FIG. 5.10 – Transformation d'une grammaire en une grammaire de chemin.

Bien que cette transformation ne préserve pas la structure du graphe engendré, la mise sous forme de grammaire de chemins préserve le caractère pondéré d'une grammaire :

si R est pondérée alors $\prec R \succ$ est pondérée.

Lorsque l'on étudie les propriétés des langages réguliers par la manipulation d'automates finis, on est amené à considérer des graphes non-déterministes. Nous avons vu précédemment que contrairement aux graphes finis, les graphes réguliers ne sont pas clos par l'opération usuelle de déterminisation *Det*. Néanmoins nous montrons comment l'utiliser pour construire, à partir d'une grammaire pondérée R, une grammaire qui engendre un graphe déterministe et qui est bi-synchronisée par R.

Lemme 5.3.8. *Tout grammaire pondérée peut être transformée en une grammaire bi-synchronisée, acceptant le même langage et engendrant un graphe déterministe.*

Démonstration :
Soit R une grammaire pondérée. Par le lemme 5.3.7, on peut considérer que R est une grammaire de chemins.
Considérons l'ensemble $N_R \cap F_2$ des parties des non-terminaux d'arité 2 de R muni d'un ordre tel que $\emptyset$ soit le plus petit élément. Soit $N' = N_R - \{Z\}$. A chaque partie $P \subseteq N'$ différente de l'ensemble vide, on note $P_2 = P \cap F_2$ et on associe un nouveau symbole $P' \in F_{2^{|P_2|}}$ et un hyperarc $<P> \ = \ P' p_1 \dots p_m$ tels que $\{p_1, \dots, p_m\} = 2^{P_2}$ et $p_1 < \dots < p_m$. Pour tout graphe G tel que $E_G \subseteq F_R - \{Z\}$, on définit l'hypergraphe

$$\begin{array}{rcl} G' & := & Det([G]) \\ & \cup & \bigcup \{ <A \in N_R \mid \exists s \in Q,\ s \xrightarrow{A} > \\ & & \qquad [Q/\emptyset, \{t \mid \exists s \in Q,\ \exists A \in P,\ s \xrightarrow[G]{A} t\}/\emptyset \neq P \subseteq N'] \\ & & \qquad\quad \mid Q \subseteq \mathcal{S}_G \ \wedge \ \exists s \in Q,\ \exists A \in N_R,\ s \xrightarrow[G]{A} \}. \end{array}$$

Pour chaque partie $P \subseteq N'$, on prend un graphe H_P tel que

$$\{\emptyset \xrightarrow{A} A \mid A \in P\} \cup \{A\emptyset \mid A \in P - F_2\} \cup \{i\emptyset\} \underset{R}{\Longrightarrow} H_P.$$

La grammaire S définie par

$$S \ := \ \{(<P>, (H_P)'[\emptyset/\{\emptyset\}] - \{i\emptyset\}) \mid P \subseteq N'\}$$

est pondérée, bi-synchronisée par R et on a $L(R) \ = \ L(S)$.

□

Prenant la grammaire de chemins de la figure 5.10 avec le nommage de sommets donné à la figure suivante. Rappelons que cette grammaire est la mise sous forme de grammaire de chemins de la grammaire pondérée de la figure 5.1.

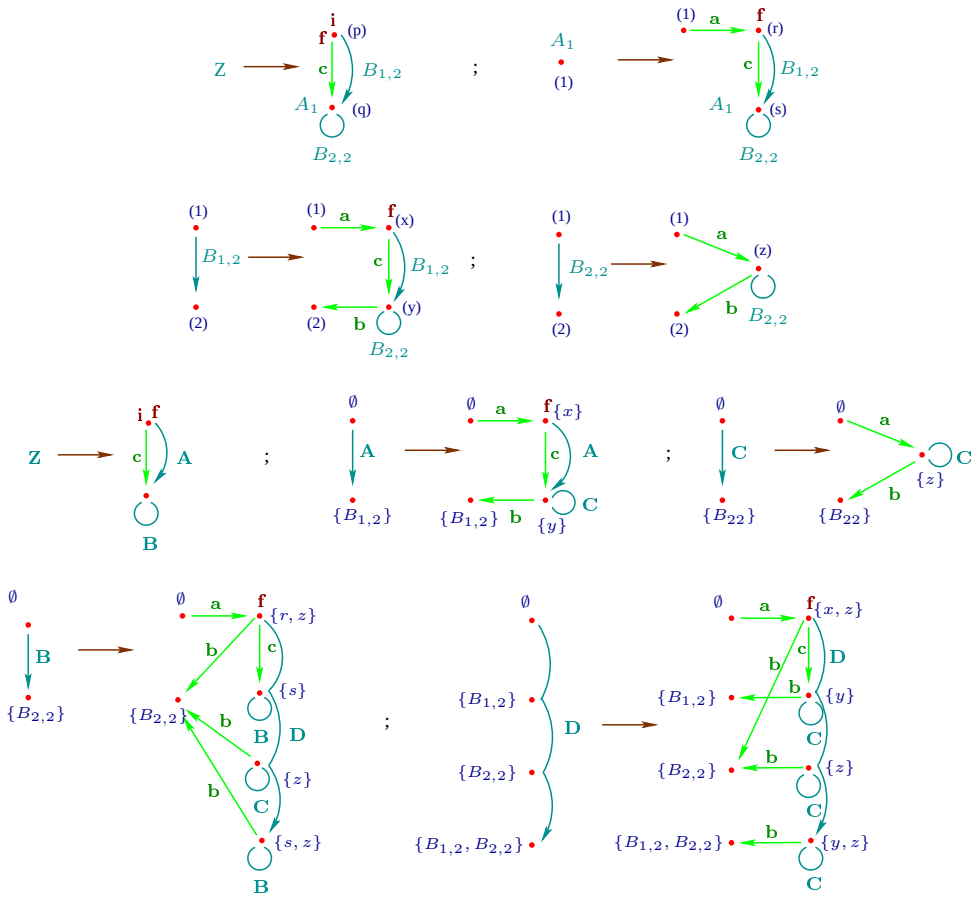


FIG. 5.11 – Déterminisation d'une grammaire de chemin.

On applique la construction du lemme 5.3.8 pour obtenir la grammaire de la figure 5.11, avec $A = \{B_{1,2}\}'$, $B = \{A_1, B_{2,2}\}'$, $C = \{B_{2,2}\}'$, $D = \{B_{2,2}, B_{1,2}\}'$ et $\{B_{1,2}\} < \{B_{2,2}\} < \{B_{1,2}, B_{2,2}\}$.

En se restreignant aux sommets accessibles à partir de i, on obtient la grammaire de la figure 5.12.

Cette grammaire engendre bien un graphe déterministe que nous représentons à la figure 5.13. Le graphe engendré par la grammaire pondérée de la figure 5.1 est représenté à la figure 5.2. Ces deux graphes acceptent le même langage et on peut remarquer de plus que les grammaires engendrant ces graphes sont bisynchronisées.

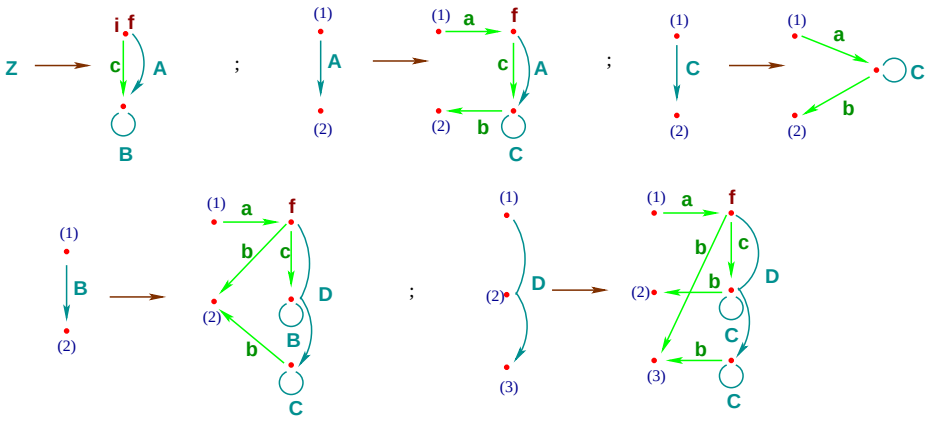


FIG. 5.12 – Grammaire engendrant un graphe déterministe acceptant le même langage que la grammaire pondérée de la figure 5.1.

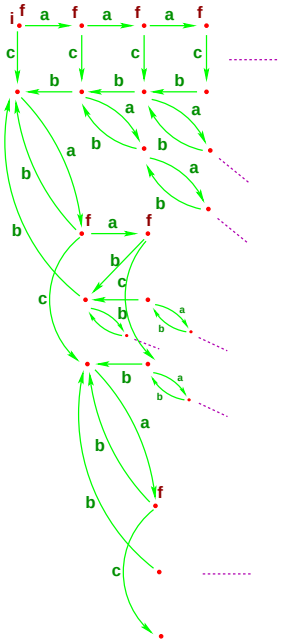


FIG. 5.13 – Graphe engendré par la grammaire 5.12.

Munis de la relation de synchronisation de grammaires, nous nous intéressons aux langages acceptés par les grammaires synchronisées. Le premier constat est que, comme les grammaires pondérées sont déterminisables et que toute grammaire engendrant un graphe déterministe est pondérée, les langages acceptés par les grammaires pondérées sont les langages algébriques

déterministes. Dans le chapitre suivant, nous définissons les langages synchronisés par une grammaire pondérée et nous montrons qu'ils forment une algèbre booléenne.

5.4 Les langages synchronisés

Nous allons maintenant nous intéresser aux langages des grammaires que l'on met en relation par synchronisation. Soient deux grammaires R et S telles que $R \triangleright S$. On dit alors que le langage de S restreint aux mots acceptés par R est un *langage synchronisé* par R et on note $Sync(R)$ l'ensemble des langages synchronisés par R :

$$Sync(R) \ := \ \{L(S) \cap L(R) \mid R \triangleright S \ \wedge S \textit{ est pondérée}\}.$$

Par exemple, les langages a^*, $(ac^*)^*$ et $\{a^n c^m b^k \mid m, n > 0 \wedge k \leq n+1\}$ sont des langages sychronisés par la grammaire de la figure 5.3.

Nous montrons que pour une grammaire R donnée, cet ensemble de langages généralise les langages réguliers. En particulier, nous montrons que :

1. tout langage régulier inclus dans $L(R)$ est un langage synchronisé par R,
2. l'ensemble des langages synchonisés par R est clos par :
   - intersection,
   - union,
   - complémentaire par rapport à $L(R)$,
   - concaténation et étoile de Kleene (sous certaines conditions).

Commençons pas montrer le premier point cité ci-dessus.

Lemme 5.4.1. *Pour toute grammaire R et pour tout langage régulier L, $L \cap L(R) \in Sync(R)$.*

Démonstration :
Nous cherchons à construire une grammaire S synchronisée par R telle que $L(S) = L(R) \cap L$. Comme L est régulier, il existe (propriété 3.2.3) un automate K fini et déterministe reconnaissant L : $L(K, i, f) = L$.
Posons $\mathcal{S}_K = \{s_1, \ldots, s_m\}$. A chaque non-terminal $A \in N_R$, on associe un nouveau symbole A' d'arité $\rho(A) \times m$ sauf que $Z' = Z$. On construit alors une grammaire S reconnaissant le langage $L \cap L(R)$. Pour toute règle $(X, H) \in R$, on associe une nouvelle règle (X, H') telle que

$$H' = [H] \times K \cup \{A(x_1, s_1) \dots (x_1, s_m) \dots \\ (x_n, s_1) \dots (x_n, s_m) \mid A \in N_R \wedge Ax_1 \dots x_n \in H\}.$$

Cet ensemble de règles restreint par accessibilité à partir de i nous donne la grammaire S. En effet, on a $L(S) = L(R) \cap L$ et S est synchronisée par R.

□

En prenant la grammaire R de la figure 5.3 et L le langage accepté par l'automate H de la figure 3.4, la construction du lemme 5.4.1 est illustrée à la figure 5.14.

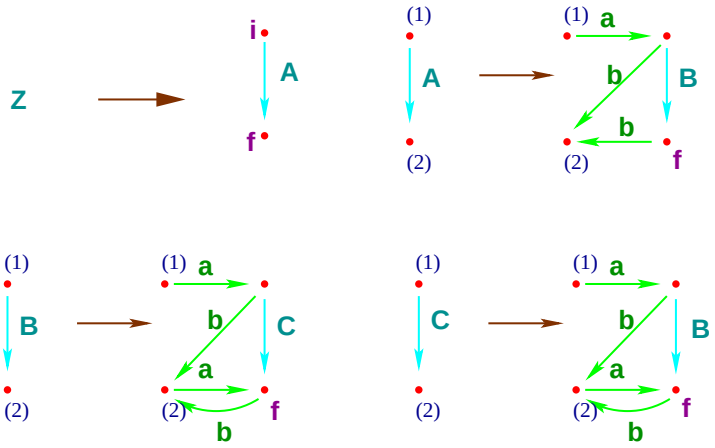


FIG. 5.14 – Les langages réguliers inclus dans $L(R)$ sont synchronisés par R.

Notons $Reg(L(R)) = \{L \in Reg(T_R^*) \mid L \subseteq L(R)\}$ l'ensemble des langages réguliers inclus dans le langage accepté par une grammaire R. Pour un synchronisateur R, tout élément de $Reg(L(R))$ est un langage synchronisé par R. De plus, par la propriété 5.3.6, cette inclusion est une égalité si R engendre un graphe fini.

Corollaire 5.4.2. *Si R engendre un graphe fini, $Sync(R) = Reg(L(R))$.*

L'intersection

Pour montrer que la classe des langages synchronisés par une grammaire R est close par intersection, nous utilisons le lemme 5.3.3.

Lemme 5.4.3. *$Sync(R)$ est clos par intersection.*

Démonstration :
Soient L_1 et L_2 deux langages de $Sync(R)$. Il existe donc deux grammaires S et T synchronisées par R telles que $L_1 = L(S) \cap L(R)$ et $L_2 = L(T) \cap L(R)$. En coloriant par f les sommets coloriés par f_1 et f_2 dans la définition de l'opération $\times_l$, on a

$$\begin{aligned} L(S \times_l T) &= L(S^\omega \times_l T^\omega) && \text{par le lemme 5.3.3} \\ &= L(S^\omega \times T^\omega) && \text{car } S, T \triangleleft R \\ &= L(S) \cap L(T) && \text{par définition de } \times . \end{aligned}$$

La clôture de $Sync(R)$ est donnée par l'équation :

$$\begin{aligned} L(S \times_l T) \cap L(R) &= (L(S) \cap L(T)) \cap L(R) \\ &= (L(S) \cap L(R)) \cap (L(T) \cap L(R)) \\ &= L_1 \cap L_2. \end{aligned}$$

□

Les grammaires S et T de la figure 5.4 sont toutes deux synchronisées par la grammaire de la figure 5.15 que l'on nomme R.

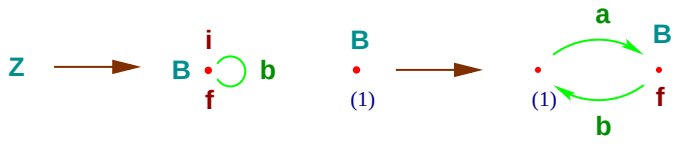


FIG. 5.15 – Un synchronisateur pour les grammaires de la figure 5.4.

Ainsi, $L(S) = \{u \mid \forall v < u,\ |v|_b \leq |v|_a \wedge |u|_b = |u|_a + 1\}$ (aussi appelé langage de Lukasiewicz) et $L(T) = \{a^n b^m \mid n \geq 0 \ \wedge \ m \geq n\}$ sont des langages synchronisés par R.
La figure 5.16 illustre la construction de l'intersection. Le langage accepté par la grammaire ainsi construite est bien $L(S) \cap L(T) = \{a^n b^{n+1} \mid n > 0\}$.

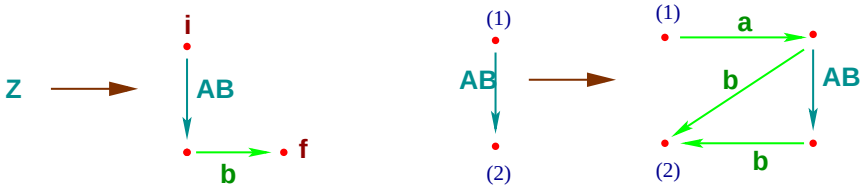


FIG. 5.16 – Intersection de langages synchronisés par utilisation du produit de synchronisation par niveau (restreint aux accessibles de i à f).

Le complémentaire

Pour construire un automate fini acceptant le complémentaire d'un langage régulier L, on considère un automate déterministe acceptant L, on le complète et on inverse les états finaux et non finaux. Concernant les langages synchronisés par une grammaire R, on va procéder de même. La différence la plus notable se situe au moment de la complémentation. En effet, dans le cas des langages réguliers, on complète par rapport à l'ensemble de tous les mots. Pour les langages synchronisés par R, si on appliquait ce principe, on perdrait la synchronisation avec R, le langage initial de la grammaire complétée pouvant contenir des mots non présent dans $L(R, i)$. Donc on ne complète que par rapport à $L(R)$.

Lemme 5.4.4. *Pour toute grammaire R pondérée, $Sync(R)$ est clos par complémentaire par rapport à $L(R)$.*

Démonstration :
Pour une grammaire R, on définit la grammaire

$$R^{-f} := \{(X, H - f\mathcal{S}_H) \mid (X, H) \in R\}$$

engendrant R^ω auquel on a retiré la couleur f. Dans ce cas, $L(R^{-f}) = \emptyset$.
De plus, on pose

$$\overline{R} := \{(X, (H - f\mathcal{S}_H) \cup \{fs \mid s \in \mathcal{S}_H - \mathcal{S}_X \wedge fs \notin H\}) \mid (X, H) \in R\}$$

la grammaire engendrant R^ω dans lequel on intervertit les sommets finaux et non finaux. Notons que ces transformations ne modifient que le coloriage et que donc les grammaires ainsi construites restent synchronisées par R. De

plus, $L(\overline{R}) = L(R,i) - L(R)$ quand R engendre un graphe déterministe.
Soit S une grammaire synchronisée par R : $R \triangleright S$. On construit la grammaire $\overline{Det(R^{-f} \otimes_l S)}$ synchronisée par R. Comme R^{-f} n'a aucun sommet final, les seules couleurs finales de $Det(R^{-f} \otimes_l S)$ sont f_2 et f^2. En identifiant ces couleurs avec f, la définition de $L(\overline{Det(R^{-f} \otimes_l S)})$ est par construction

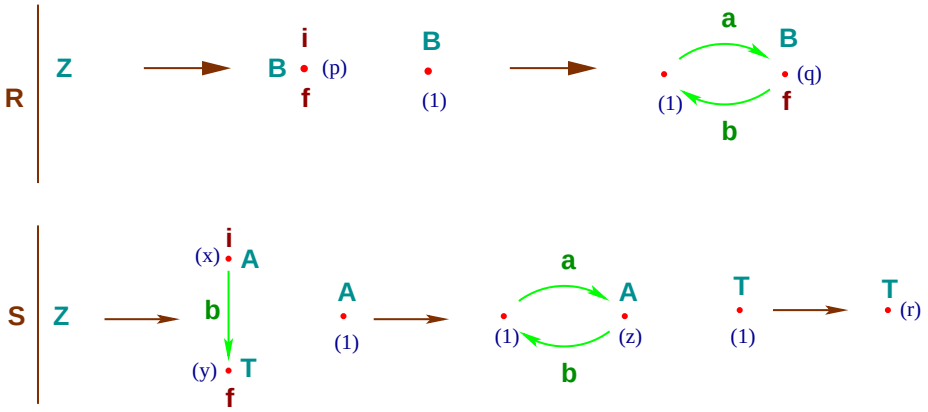
$$\begin{aligned} L(\overline{Det(R^{-f} \otimes_l S)}) &= L(Det(R^{-f} \otimes_l S), i) - L(Det(R^{-f} \otimes_l S)) \\ &= L(R^{-f} \otimes_l S, i) - L(R^{-f} \otimes_l S) \\ &= (L(R,i) \cup L(S,i)) - L(S) \\ &= L(R,i) - L(S). \end{aligned}$$

La clôture de $Sync(R)$ par complément (par rapport à $L(R)$) est alors donnée par :

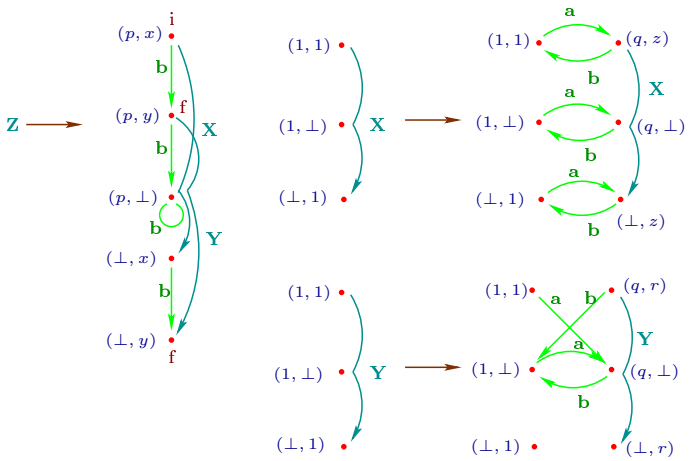
$$\begin{aligned} L(R) - L(S) &= (L(R) \cap L(R,i)) - L(S) \\ &= L(R) \cap (L(R,i) - L(S)) \\ &= L(R) \cap L(\overline{Det(R^{-f} \otimes_l S)}). \end{aligned}$$

□

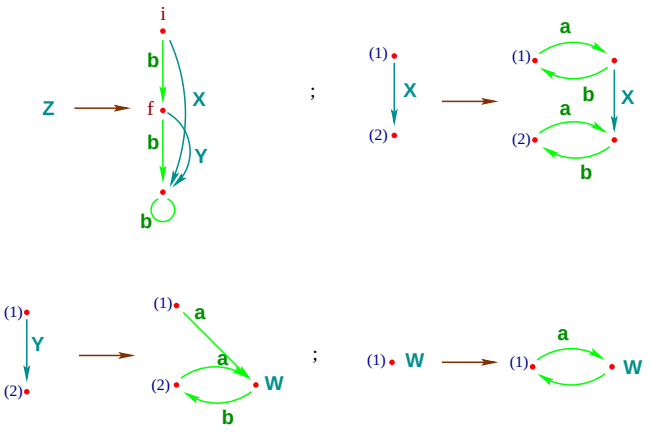
Appliquons la construction aux grammaires R et S suivantes :



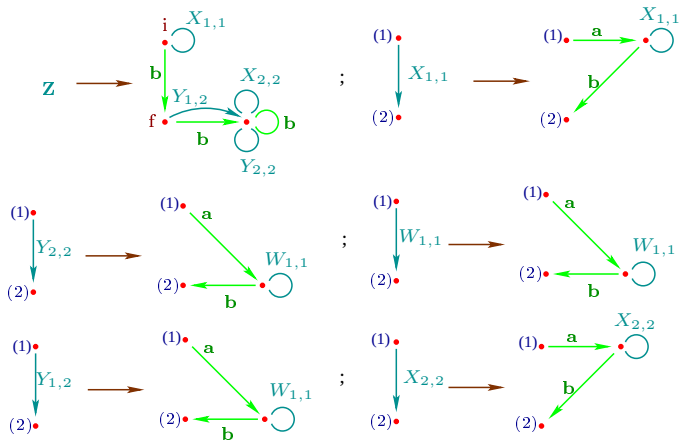
Dans la grammaire S, on a colorié le sommet y par un non-terminal inutile T afin que tout sommet non d'entrée est sommet de sortie.On construit la grammaire $R^{-f} \otimes_l S$ représentée ci-après (où f remplace f_2 et f^2 dans la définition de $\otimes_l$) pour $X = [B, A, \{(1,1)\}]$ et $Y = [B, T, \{(1,1)\}]$:



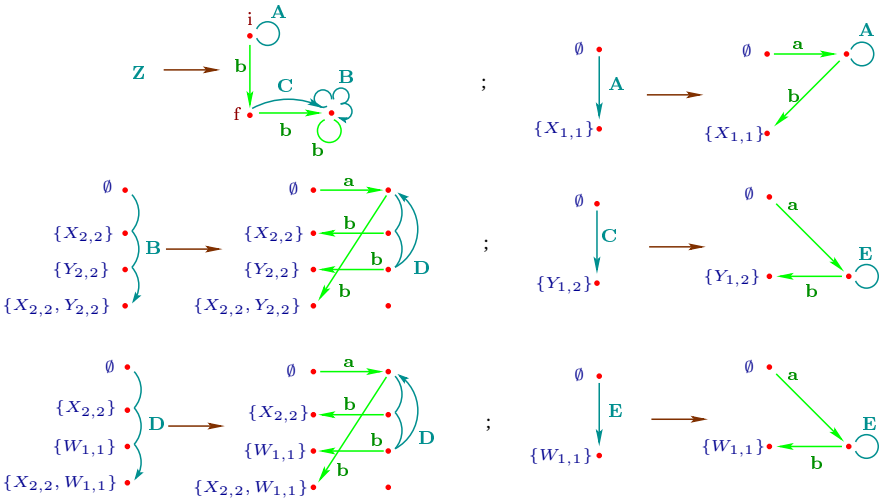
que l'on peut restreindre par accessibilité à partir de i en la grammaire :



Le calcul du complémentaire d'un langage régulier requiert de considérer des graphes finis déterministes. Nous étendons la construction vue dans le paragraphe 2.2.3. Pour déterminiser la grammaire précédente, on va lui appliquer la construction du lemme 5.3.8. Pour cela, considérons la grammaire de chemins ci-dessous obtenue par le lemme 5.3.7 :



Par le lemme 5.3.8, la grammaire de la figure suivante (avec $A = \{X_{1,1}\}'$, $B = \{X_{2,2}, Y_{2,2}\}'$, $C = \{Y_{1,2}\}'$, $D = \{X_{2,2}, W_{1,1}\}'$ et $E = \{W_{1,1}\}'$) engendre un graphe déterministe et accepte $L(R^{-f} \otimes_l S)$.



Imitant les opérations sur les graphes finis, on échange les sommets finaux avec les non-finaux pour obtenir la grammaire $\overline{Det(R^{-f} \otimes_l S)}$ dont la restriction aux sommets accessibles de i est présentée à la figure 5.17. Comme $L(R, i) = \Sigma^*$, cette grammaire accepte le complémentaire du langage de Lukiasiewicz.

L'union

Par clôture par complément et intersection, on obtient la clôture par union. Nous donnons ici une construction directe.

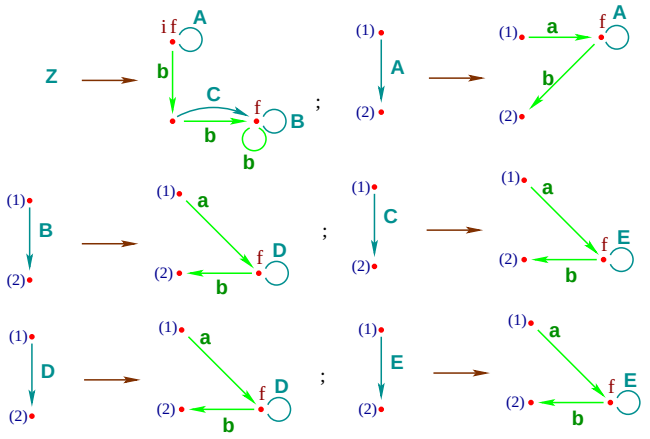


FIG. 5.17 – Grammaire acceptant le complémentaire du langage de Lukasiewicz.

Lemme 5.4.5. *$Sync(R)$ est clos par union.*

Démonstration :
Prenons L_1 et L_2 deux langages de $Sync(R)$. Soit A_1 (resp. A_2) une grammaire synchronisée par R telle que $L_1 = L(A_1) \cap L(R)$ (resp. $L_2 = L(A_2) \cap L(R)$). Par construction, on a $L(A_1 \otimes A_2) = L(A_1) \cup L(A_2)$ en identifiant $f_1,\ f^1,\ f_2,\ f^2$ à f. La clôture par union est donnée par :

$$\begin{aligned} L_1 \cup L_2 &= (L(A_1) \cap L(R)) \cup (L(A_2) \cap L(R)) \\ &= (L(A_1) \cup L(A_2)) \cap L(R) \\ &= L(A_1 \otimes A_2) \cap L(R). \end{aligned}$$

□

La grammaire de la figure 5.18 est obtenue à partir de la grammaire $S \otimes_l T$ (présentée à la figure 5.6 par restriction aux accessibles par i. Pour H cette grammaire, on a $L(H_{\{f,f_1,f_2,f^1,f^2\}}) = L(S) \cup L(T)$ pour les grammaires S et T de la figure 5.4.

Par les lemmes 5.4.3, 5.4.4 et 5.3.8, on obtient que, pour une grammaire R fixée, $Sync(R)$ est une algèbre de Boole de langages algébriques déterministes.

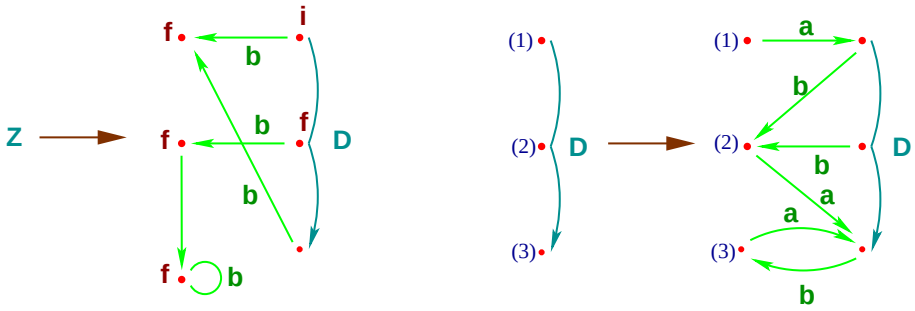


FIG. 5.18 – Grammaire acceptant l'union des grammaires S et T de la figure 5.4.

Théorème 5.4.6. *Pour toute grammaire pondérée R, $Sync(R)$ est de façon effective une algèbre de Boole (relativement à $L(R)$) de langages algébriques déterministes et contenant les langages réguliers inclus dans $L(R)$.*

Dans [AM04], Alur et Madhusudan montrent que les classes de langages qu'ils définissent sont également closes par concaténation. Dans le cas présent, nous présentons une grammaire R dont l'ensemble $Sync(R)$ des langages synchronisés par R n'est pas clos par concaténation. Pour cela, il suffit de considérer comme synchronisateur une grammaire R dont le langage $L(R)$ n'est pas clos par concaténation. Pour R la grammaire de la figure 5.3, le mot *abab* est un mot de $L(R).L(R)$ mais pas un mot de $L(R)$.

Nous pouvons donner des conditions suffisantes sur un synchronisateur pour que la classe des langages qu'il synchronise soit close par concaténation et étoile de la concaténation.

La concaténation

Une condition naturelle pour que l'ensemble $Sync(R)$ des langages synchronisés par une grammaire R soit clos par concaténation consiste à dire que le langage $L(R)$ accepté par R soit clos par concaténation avec lui même : $L(R).L(R) \subseteq L(R)$. Cette condition est évidemment nécessaire mais elle n'est pas suffisante. En effet considérons la grammaire R ci-dessous acceptant le langage $L(R) = \{a^n b^n \mid n \geq 1\}(a+b)^*$.

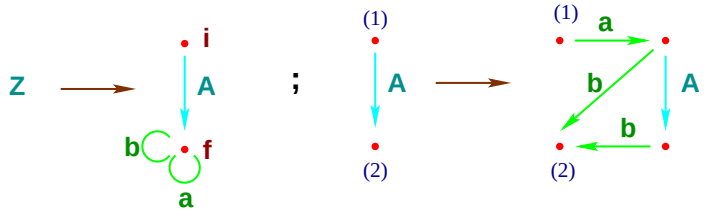


FIG. 5.19 – R telle que $Sync(R)$ n'est pas clos par concaténation.

On a bien $L(R).L(R) \subseteq L(R)$. De plus, la grammaire acceptant le langage $\{a^n b^n \mid n \geq 1\}$ (cf. figure 4.6) est synchronisée par R mais la concaténation de ce langage par lui même, *i.e.* $\{a^n b^n a^m b^m \mid n, m \geq 1\}$, n'est pas inclus dans $Sync(R)$. Néanmoins, si l'on considère des contraintes plus fortes que la simple clôture par concaténation, on peut montrer que l'ensemble des langages synchronisés par une grammaire est clos par concaténation, simplement en étendant les constructions existantes pour les automates finis.

Une grammaire R est *cyclique* si elle engendre un graphe G déterministe et que l'unique sommet initial de G est son unique sommet final : $is \in G \iff fs \in G$.
Remarquons que pour tous mots $u, v \in L(R)$, $u.v \in L(R)$ et $||u.v||_R = ||u||_R + ||v||_R = 0$.

Théorème 5.4.7. *Les langages synchronisés par une grammaire cyclique sont clos par concaténation et étoile de Kleene.*

Démonstration :
Soit R une grammaire cyclique. Commençons par montrer que $Sync(R)$ est clos par concaténation. Soient $L_1 \in Sync(R)$ et $L_2 \in Sync(R)$. Soient $S \triangleleft R$ et $T \triangleleft R$ deux grammaires telles que $L_1 = L(S)$ et $L_2 = L(T)$. On peut supposer que $N_S \cap N_T = \{Z\}$ et que les graphes engendrés par R, S et T sont accessibles par i et co-accessibles par f. De plus, comme R est cyclique, les sommets finaux de S (et de T) sont des sommets de l'axiome $S(Z)$ ($T(Z)$).
Soit $Z' \in F_1$ un nouveau non-terminal. La *concaténation* $S.T$ de S par T est

la grammaire :

$$\begin{aligned} S.T \;&:=\; \{(X,(H-f\mathcal{S}_H)\;\cup\;Z'\mathcal{S}_{H,f}) \mid (X,H)\in S\} \\ &\cup\; T-\{(Z,T(Z))\}\;\cup\;\{(Z'r,T(Z)-\{ir\}) \mid ir\in T(Z)\}. \end{aligned}$$

Nous étendons donc le principe de concaténation des automates finis aux grammaires de graphes. On a donc

$$L(S.T) \;=\; L(S).L(T) \;=\; L_1.L_2 \;\subseteq L(R).$$

De plus, même si $(S.T)^\omega$ est non-déterministe, $S.T$ est une grammaire pondérée synchronisée par R. Par le lemme 5.3.8, il existe une grammaire $(S.T)_{det}$ synchronisée par R et telle que $L((S.T)_{det}) = L(S.T) = L_1.L_2$. Donc $L_1.L_2 \in Sync(R)$.

Maintenant que la clôture de $Sync(R)$ par concaténation est établie, montrons qu'il en est de même pour sa fermeture réflexive et transitive : l'étoile de Kleene. Soit $L \in Sync(R)$. Comme R est cyclique, $L^* \subseteq L(R)^* \subseteq L(R)$. Soit $S \triangleleft R$ une grammaire telle que $L = L(S)$. On construit une grammaire pondérée S^* synchronisée par R et reconnaissant $L(S)^*$. A chaque non-terminal $A \neq Z$ et à chaque $P \subseteq [\rho(A)]$, on associe un nouveau non-terminal A_P d'arité $\rho(A)+1$. On pose 0 un nouveau sommet. Les non-terminaux de la grammaire S^* sont donnés par

$$N_{S^*} \;=\; \{Z\}\;\cup\;\{A_P \mid A\in N_S-\{Z\}\;\wedge\;P\subseteq[\rho(A)]\}$$

et l'axiome de S est donné par la règle

$$\begin{aligned} Z \;\to\; & [S(Z)]\;\cup\;\{s\xrightarrow{a}r \mid ir\in S(Z)\;\wedge\exists t,\; s\underset{[S(Z)]}{\xrightarrow{a}}t\;\wedge\;ft\in S(Z)\} \\ \cup\; & \{A_{\{n\;\mid\;fa_n\in S(Z)\}}ra_1\dots a_{\rho(A)} \mid ir\in S(Z) \\ & \qquad\qquad \wedge\, Aa_1\dots a_{\rho(A)}\in S(Z)\;\wedge\;A\in N_S\}. \end{aligned}$$

Pour toute règle $(A1\dots\rho(A), S(A))$ et toute partie $P\subseteq[\rho(A)]$, on définit la règle :

$$\begin{aligned} A_P01\dots\rho(A) \;\to\; & [S(A)]\;\cup\;\{s\xrightarrow{a}0 \mid \exists p\in P,\; s\underset{[S(A)]}{\xrightarrow{a}}p\} \\ \cup\; & \{B_{\{n\;\mid\;b_n\in P\}}0b_1\dots b_{\rho(B)} \mid Bb_1\dots b_{\rho(B)}\in S(A)\;\wedge B\in N_S\}. \end{aligned}$$

Ainsi, $L(S^*) = (L(S))^* = L^*$ et S^* est une grammaire pondérée synchronisée par $R : S^* \triangleleft R$. Par le lemme 5.3.8, il existe une grammaire S' bi-synchronisée avec S^* telle que S'^{ω} est déterministe et $L(S') = L(S^*) = L^*$. Ainsi $L^* \in Sync(R)$, c'est-à-dire que $Sync(R)$ est clos par étoile de Kleene.

□

Le graphe présenté à la figure 5.20 n'est pas le graphe d'une grammaire cyclique mais la famille des langages qu'il synchronise est close par concaténation (nous discuterons de ceci dans la partie suivante). Nous cherchons donc

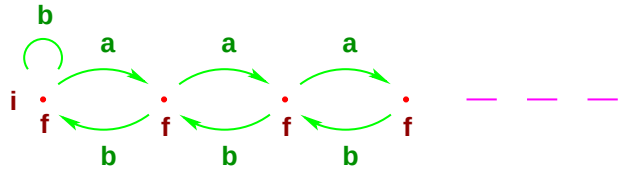


FIG. 5.20 – Un synchonisateur pour les langages visibles.

à étendre le critère de cyclicité à la recherche d'un critère moins restrictif pour la clôture par concaténation.

Une grammaire R est *itérative* si R^ω est déterministe, que $L(R)$ est clos par concaténation et que

$$\forall u \in L(R),\ \forall v \in L(R,i), u.v \in L(R,i)\ \wedge\ ||uv||_R\ =\ ||u||_R\ +\ ||v||_R.$$

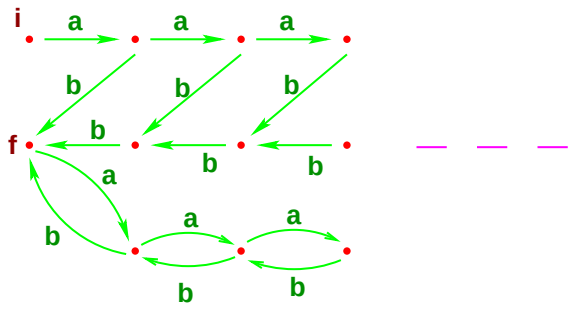


FIG. 5.21 – Un graphe de grammaire itérative non cyclique.

En suivant la preuve du théorème 5.4.7, on montre que la famille des langages synchronisés par une grammaire itérative est close par concaténation et étoile. Néanmoins il n'est pas nécessaire qu'une grammaire soit itérative pour que l'ensemble des langages qu'elle synchronise soit close par concaténation. En effet, la grammaire

$$R = \{(Z, \{bx, ix, fx, Ax\}), (A1, \{a1y, by1, fy, Ay\})\}$$

qui engendre le graphe 5.20 n'est pas itérative mais la famille des langages synchronisés par cette grammaire forme exactement les langages visibles ([AM04]) définis sur les alphabets $\Sigma_{push} = \{a\}$ et $\Sigma_{pop} = \{b\}$. Dans la partie suivante, nous discutons de la relation entre les langages synchronisés et les familles de langages définis dans [AM04] et [NS07].

Langages synchronisés et autres formalismes

Une motivation de notre travail fut de généraliser les travaux de [AM04] et de [NS07].
Dans un premier temps, nous décrivons les langages visibles comme des langages synchronisés par un automate régulier. Soit $\Sigma = \Sigma_{push} \uplus \Sigma_{pop} \uplus \Sigma_{int}$

une partition de l'alphabet Σ. L'ensemble des langages visibles sur Σ est l'ensemble des langages synchronisés par la grammaire VPL présentée à la figure 5.22 où $p \xrightarrow{\Sigma} q \;=\; \{p \xrightarrow{a} q \mid a \in \Sigma\}$.

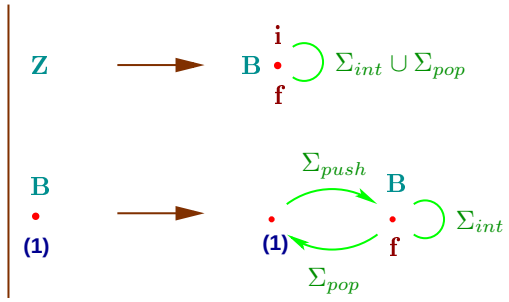


FIG. 5.22 – La grammaire VPL synchronisant les langages visibles.

Proposition 5.4.8. *Les langages synchronisés par la grammaire VPL sont les langages visibles.*

Démonstration :

i) Commençons par vérifier que tout langage visible L accepté par un automate visible M (sur la partition $\Sigma = \Sigma_{push} \uplus \Sigma_{pop} \uplus \Sigma_{int}$) est un langage synchronisé par la grammaire VPL.

D'après le lemme 4.2.5, il existe une grammaire S engendrant G_M par longueur (de sommets) croissante à partir de 2 :

$$l^S_{G_M}(s) = |s| - 2 \text{ pour tout } s \in \mathcal{S}_{G_M}.$$

Comme M est visible, $S \triangleleft VPL$ donc $L = L(S) \in Sync(VPL)$.

ii) Vérifions maintenant que toute grammaire S synchronisée par VPL accepte un langage L visible.

La réciproque du lemme 4.2.5 (donnée dans [Ca07]) construit un automate à pile M tel que

$$G_M \in S^\omega \text{ et } \forall s \in \mathcal{S}_{G_M},\ |s| = l^S_{G_M}(s) + 2.$$

Comme $S \triangleleft VPL$, M est visible.

□

Nous avons aussi rappelé (cf. chapitre 4.1) la synchronisation par transducteurs finis donnée dans [Ca06]. L'ensemble des langages synchronisés par un transducteur fini peut être obtenu par synchonisation avec une grammaire linéaire. Une grammaire est *linéaire* si pour chaque règle, le membre droit ne contient qu'au plus un hyperarc non-terminal.

Proposition 5.4.9. *Pour tout transducteur M, on peut construire une grammaire linéaire R telle que R^ω soit déterministe et $Sync(R) = Sync(M)$.*

Démonstration :
Soient $m = max\{|x| \mid T \times \{x\} \cap F_M \neq \emptyset\}$ la valeur absolue maximale des entiers des étiquettes de M et $n = \lfloor \frac{m}{2} \rfloor + 1$.
Le graphe défini par

$$\begin{aligned} \vec{M} &:= \{(p,j) \xrightarrow{a} (q, j+x) \mid p \xrightarrow[M]{(a,x)} q \wedge j \in \mathbb{Z}\} \\ &\cup \{i(p,0) \mid ip \in M\} \cup \{f(p,j) \mid p \in \mathcal{S}_M \wedge j \in \mathbb{Z}\} \end{aligned}$$

est synchronisé par M et $L(\vec{M}, i) = Dom(L(M,i))$. On engendre $\vec{M}$ par sommets (p,j) selon $|j|$ croissant à l'aide de la grammaire linéaire R :

$$\left|\begin{array}{rcll} Z & \rightarrow & P_0 & \cup \{Z_1\} \\ Z_1 & \rightarrow & P_1 & \cup \{Z_2\} \\ & \vdots & & \\ Z_{n-1} & \rightarrow & P_{n-1} & \cup \{Z_n\} \\ Z_n & \rightarrow & P_n & \cup \{Z_{n+1}\} \end{array}\right.$$

avec les non-terminaux satisfaisant

$$(Z \neq) Z_1(1) \neq \ldots \neq Z_n(1) = Z_{n+1}(1)$$

et pour chaque $0 \leq l \leq n$, les arcs terminaux présents dans le membre droit de Z_l est l'ensemble

$$\begin{aligned} P_l &:= \{(p,j) \xrightarrow{a} (q,k) \mid p \xrightarrow[M]{(a,k-j)} q \wedge max\{|j|,|k|\} = l\} \\ &\cup \{f(p,l) \mid fp \in \mathcal{S}_M\} \end{aligned}$$

et l'ensemble des sommets de l'hyperarc non-terminal Z_l est

$$\mathcal{S}_{Z_l} = \mathcal{S}_M \times \{l\} \cup \{(p,j) \mid j < l \wedge \exists (q,k) \underset{\vec{M}}{\leftrightarrow} (p,j),\ |k| > l\}.$$

Pour tout $2 \leq j \leq |Z_n|$ et pour $Z_n(j) = (p,x)$, on a

$$Z_{n+1}(j) = \begin{cases} (p, x+1) & \text{si} \quad x \geq 0 \\ (p, x-1) & \text{si} \quad x < 0. \end{cases}$$

Ainsi, R engendre $\vec{M}$ avec $||(p,j)||_R = |j|$, et donc $Sync(R) = Sync(M)$.

□

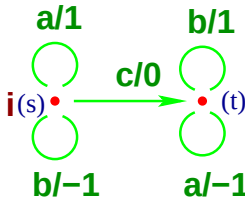


FIG. 5.23 – Un transducteur fini.

Prenons par exemple le transducteur fini ci-dessus. Par la proposition 5.4.9, on construit la grammaire représentée à la figure 5.24.

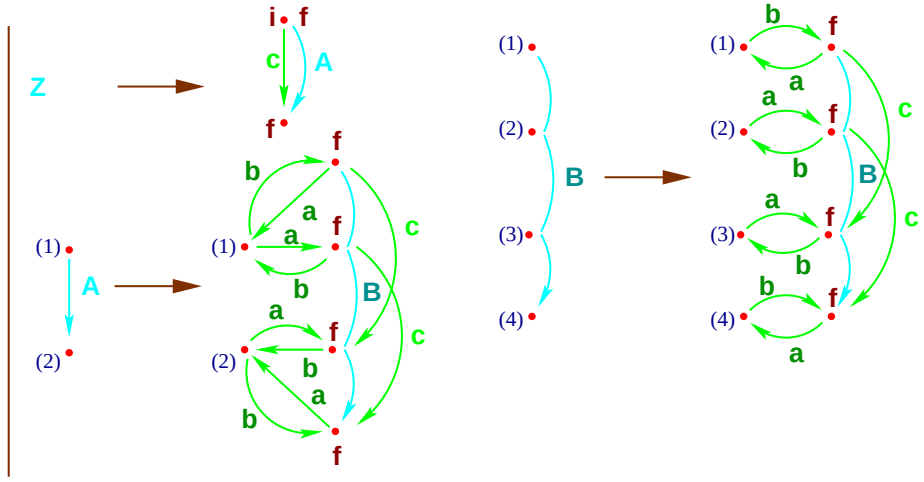


FIG. 5.24 – Grammaire synchronisant les mêmes langages que le transducteur de la figure 5.23.

Nous comparons également la synchronisation de grammaires avec la synchronisation des automates à hauteur de pile déterministe [NS07].
Un *automate à hauteur de pile déterministe* est un automate à pile complet satisfaisant les conditions suivantes :

$$is_0 \in M \ \wedge \ s_0 \overset{u}{\underset{G_M}{\Longrightarrow}} px \ \wedge \ s_0 \overset{u}{\underset{G_M}{\Longrightarrow}} qy \ \Longrightarrow \ |x| = |y|$$

Nowotka et Srba ont défini la synchronisation de deux automates à hauteur de pile déterministe M et M' par

$$is_0 \in M \ \wedge \ is'_0 \in M' \ \wedge \ s_0 \overset{u}{\underset{G_M}{\Longrightarrow}} px \ \wedge \ s'_0 \overset{u}{\underset{G_{M'}}{\Longrightarrow}} qy \ \Longrightarrow \ |x| = |y|$$

et la famille de langages synchronisés par un automate M par

$$Sync(M) \ := \ \{L(M') \mid M' \text{ synchronisé par } M\}.$$

La synchronisation d'automates à hauteur de pile déterministe correspond à la synchronisation de grammaires.

Proposition 5.4.10. *On peut transformer tout automate à hauteur de pile déterministe M en une grammaire de graphes R tel que R^ω soit déterministe, de degré fini et $Sync(R) = Sync(M)$.*

Démonstration :
Par la propriété 4.2.6, il suffit de transformer M en une grammaire pondérée R tel que R^ω est de degré fini et $Sync(R) = Sync(M)$. Le théorème 5.6 de [Ca07] stipule que toute restriction régulière d'un graphe préfixe est régulier par longueur. On peut donc transformer M en une grammaire déterministe de graphes R_M engendrant G_M par longueur croissante.
En notant tout sommet non entrée de R_M par f, on obtient la grammaire suivante :

$$\overline{R_M} := \{(X, H \cup \{fc \mid c \in \mathcal{S}_H - \mathcal{S}_X\}) \mid (X, H) \in R_M\}.$$

Comme M est complète, $L(\overline{R_M}) = \Sigma^*$. De plus, comme M est à hauteur de pile déterministe, $\overline{R_M}$ est une grammaire pondérée.
Il reste à prouver que $Sync(\overline{R_M}) = Sync(M)$.
Vérifions que $Sync(M) \subseteq Sync(\overline{R_M})$.
Soit $L \in Sync(M)$ *i.e.* $L = L(M')$ pour un automate à hauteur de pile déterministe M' synchronisé par M. Comme $R_{M'}$ est une grammaire pondérée engendrant $G_{M'}$ par longueur croissante, $R_{M'} \bowtie \overline{R_M}$. Ainsi,

$$L = L(M') = L(R_{M'}) = L(R_{M'}) \cap \Sigma^* = L(R_{M'}) \cap L(\overline{R_M}) \in Sync(\overline{R_M}).$$

Vérifions que $Sync(\overline{R_M}) \subseteq Sync(M)$.
Soit $L \in Sync(\overline{R_M})$ *i.e.* $L = L(R')$ pour une grammaire pondérée $R' \triangleleft R_M$. On suppose que $L(R', i) = L(G_M, i) = \Sigma^*$. Si ce n'est pas le cas, on remplace R' par $R' \otimes R_M$. Par le lemme 5.3.8 et la propriété 4.2.6, on considère que R'^ω est un graphe déterministe dont tout sommet est accessible à partir de i.
Comme $G_M \in R_S^\omega$, le graphe R_M^ω est de degré fini et par le lemme 5.3.1, R'^ω est également de degré fini.

Par le lemme 5.7 de [Ca07], on peut transformer R' en un système de récriture de mots M' dont le graphe de transition préfixe $Pre(M')$ restreint aux accessibles à partir de i est égal à $\overline{Gen(R')}$ obtenu à partir du graphe canonique $Gen(R')$ en remplaçant chaque sommet pAU par $(p, A)U\bot$. Comme $L(R', i) = \Sigma^*$, l'automate S' est complet. Et comme R'^ω est déterministe, S' est un automate à hauteur de pile déterministe. Nommant x le sommet de l'axiome de R' étiqueté par i, on prend pour configuration initiale

$c_0 = (x, Z)\bot$. Ainsi, $G_{S'} = \overline{Gen(R')}$ et donc S' est synchronisé par S. Et finalement on a

$$L = L(R') = L(G_{S'}) = L(S') \in Sync(S).$$

□

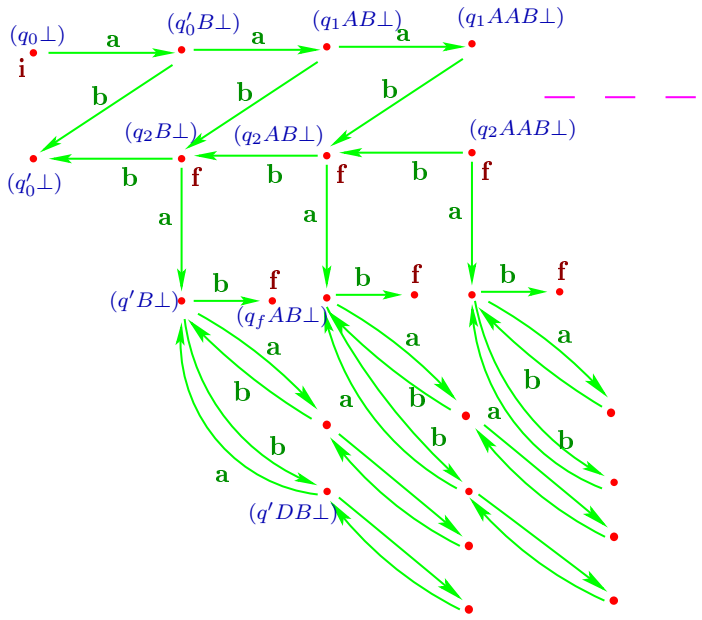
Prenons comme exemple l'automate à hauteur de pile déterministe $M = (\{q_0, q_1, q_2, q_3, q', q_f\}, \{A, B, C, D\}, \{a, b\}, q_0, \{q_3, q_f\}\}, \delta)$ avec δ défini par

$q_0\bot$	$\xrightarrow{a}$	q_1B	q_1B	$\xrightarrow{a}$	q_2AB	q_3B	$\xrightarrow{b}$	q_1
q_1B	$\xrightarrow{b}$	q_1	q_2A	$\xrightarrow{b}$	q_3	q_3A	$\xrightarrow{b}$	q_3
q_2A	$\xrightarrow{a}$	q_2AA	q_3	$\xrightarrow{a}$	q'	$q'A$	$\xrightarrow{b}$	q_fAA
$q'B$	$\xrightarrow{b}$	q_fAB	$q'A$	$\xrightarrow{a}$	$q'CA$	$q'A$	$\xrightarrow{b}$	$q'DA$
$q'B$	$\xrightarrow{a}$	$q'CB$	$q'B$	$\xrightarrow{b}$	$q'DB$	$q'D$	$\xrightarrow{b}$	$q'DD$
$q'D$	$\xrightarrow{a}$	q'	$q'C$	$\xrightarrow{a}$	$q'CC$	$q'C$	$\xrightarrow{b}$	q'

qui reconnaît le langage (cf. théorème 8 [NS07])

$$L_3 = \{a^m b^n w \mid m > n > 0,\ |w|_a = |w|_b,\ w(1) = a \text{ si } w \neq \varepsilon\}$$

et dont le graphe de transitions accessibles à partir de i est le suivant :



Il est engendré par longueur croissante par la grammaire à la figure 5.25 qui définit la même classe de langages synchronisés que M.

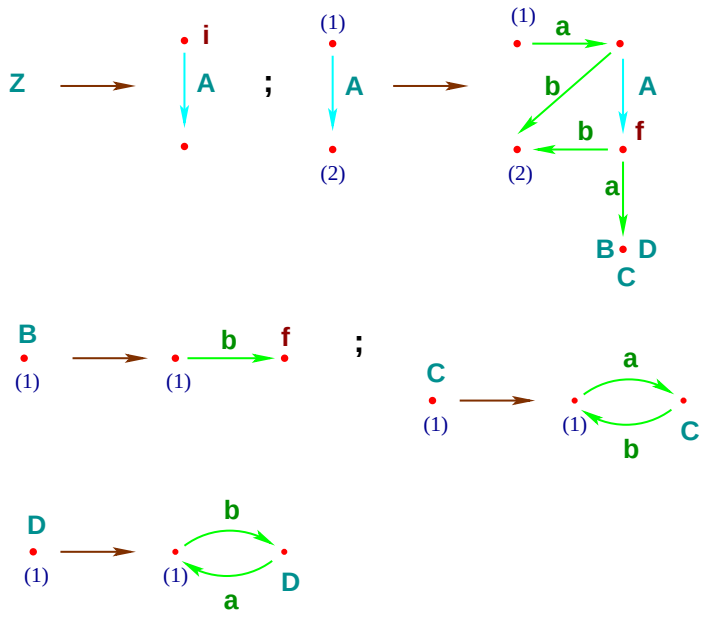


FIG. 5.25 – Automate à hauteur de pile en grammaire : la grammaire.

La synchronisation permet par ailleurs de décrire des sous-familles connues de langages algébriques. Par exemple, on considère la famille des langages équilibrés [BB02] sur un alphabet $\Sigma = A \uplus B \uplus \overline{A}$ avec $A = \{a_1, \ldots, a_n\}$ les symboles d'empilement, $\overline{A} = \{\overline{a_1}, \ldots, \overline{a_n}\}$ les symboles de dépilement et B les symboles d'action interne. Ces langages sont les langages synchronisés par la grammaire engendrant le graphe de la figure 5.26.

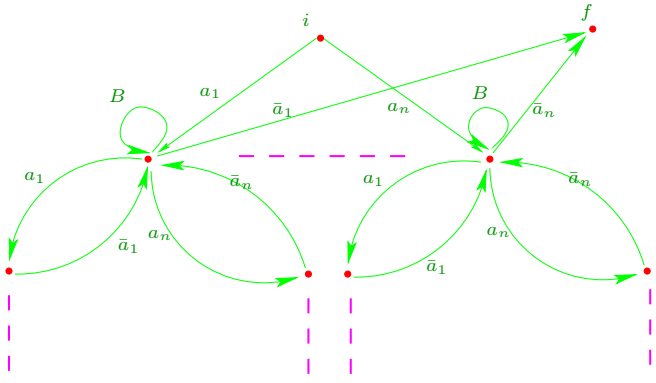


FIG. 5.26 – Graphe d'un synchronisateur pour les langages équilibrés.

Chapitre 6

Une extension

Jusqu'alors, les algèbres de Boole que nous définissions étaient des sous-ensembles de langages algébriques déterministes. En considérant des synchronisateurs acceptant des langages non déterministes, on étend les propriétés de la synchronisation à des ensembles de langages algébriques non-ambigus pour lesquels l'inclusion est décidable ([AN00]).

6.1 Les langages non-ambigus

Une grammaire algébrique R est *non-ambiguë* si pour tout mot de $L(R)$ il n'existe qu'une dérivation gauche l'acceptant, ou de manière équivalente qu'un seul arbre de dérivation. Ainsi, toute grammaire algébrique déterministe est non-ambiguë.
Un langage algébrique L est *non-ambigu* s'il est engendré par une grammaire algébrique non-ambiguë. De plus, on dit qu'un langage est *intrinsèquement ambigu* s'il n'existe pas de grammaire non-ambiguë l'acceptant.
Par exemple, la grammaire d'axiome S donnée à la figure 6.1 est ambiguë puisqu'il existe deux manières d'engendrer le mot ab. Néanmoins le langage que cette grammaire accepte, $\{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0\}$, n'est pas intrinséquement ambigu, puisqu'il est accepté par la grammaire non-ambiguë de la figure 6.2.

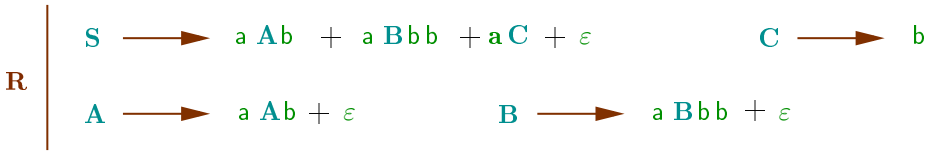


FIG. 6.1 – Une grammaire algébrique ambiguë.

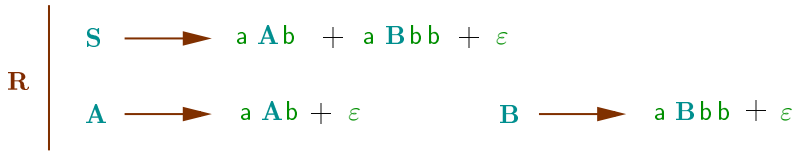


FIG. 6.2 – Une grammaire algébrique non-ambiguë.

6.2 Les grammaires non-ambiguës

On étend la notion de non-ambiguïté aux grammaires de graphes. Un graphe G est *non-ambigu* si deux chemins acceptant ont des étiquettes différentes :

$$\left.\begin{array}{r} s_0 \xrightarrow[G]{a_1} s_1 \ldots \xrightarrow[G]{a_n} s_n \\ t_0 \xrightarrow[G]{a_1} t_1 \ldots \xrightarrow[G]{a_n} t_n \\ is_0,\ it_0,\ fs_n,\ ft_n \in G \end{array}\right\} \implies s_i = t_i\ \forall i \in [0,n].$$

Par abus de langage, on appelle grammaire de graphe non-ambiguë toute grammaire engendrant un graphe non-ambigu.
La grammaire de la figure 6.3 est non-ambiguë et reconnaît le même langage que la grammaire algébrique de la figure précédente.

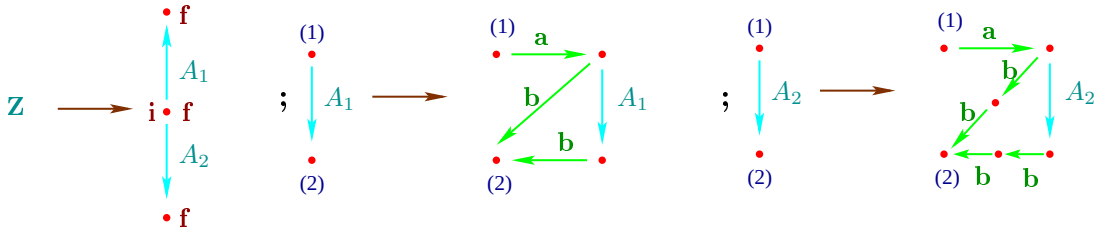


FIG. 6.3 – Une grammaire de graphe non-ambiguë.

Nous avions vu précédemment comment restreindre une grammaire (engendrant un graphe déterministe) de sorte que le sommet étiqueté par la

couleur initiale soit un sommet de l'axiome (5.3.2). Nous présentons ici une transformation qui nous permet de ne considérer que des grammaires dont les sommets initaux et finaux appartiennent à l'axiome.
Une grammaire R est *initiale* si les seuls sommets étiquetés par i ou f appartiennent à son axiome :

$$(X, H) \in R \ \wedge \ X \neq Z \implies \mathcal{S}_{H,i} = \emptyset = \mathcal{S}_{H,f}.$$

Soient $\iota, \phi \in F_2$ deux nouvelles étiquettes d'arcs ($\iota, \phi \notin F_R$). Et on considère de plus deux nouveaux sommets ι et ϕ qui n'appartiennent pas à R.

Proposition 6.2.1. *On peut transformer toute grammaire R en une grammaire initiale $[R, \iota, \phi]$ telle que $L([R, \iota, \phi]) = \iota L(R)\phi$.*

Démonstration :
Soit R une grammaire de graphes et deux nouveaux symboles ι et ϕ.
A chaque non-terminal $A \in N_R - \{Z\}$, on associe un nouveau symbole $A_{\iota,\phi}$ d'arité $\rho(A) + 2$. La grammaire $[R, \iota, \phi]$ est construite comme suit :

$$\begin{array}{rcl} [R, \iota, \phi] & := & \{(Z, H_{\iota,\phi} \cup \{i\iota, f\phi\}) \mid (Z, H) \in R\} \\ & \cup & \{(A_{\iota,\phi} X \iota \phi, H_{\iota,\phi}) \mid (AX, H) \in R \wedge A \neq Z\} \end{array}$$

où $H_{\iota,\phi}$ est le graphe obtenu à partir H en effectuant les opérations suivantes :

$$\begin{array}{rcl} H_{\iota,\phi} & := & [H] - \{i, f\}\mathcal{S}_H \quad \cup \quad \{A_{\iota,\phi} X \iota \phi \mid AX \in H \wedge A \in N_R\} \\ & \cup & \{\iota \xrightarrow{\iota} s \mid is \in H\} \quad \cup \quad \{s \xrightarrow{\phi} \phi \mid fs \in H\}. \end{array}$$

La grammaire $[R, \iota, \phi]$ est bien initiale et reconnaît $\iota L(R)\phi$.

□

Comme pour le cas déterministe, on définit la relation de synchronisation par niveau. Une grammaire R est *non-ambiguë par niveau* si pour tout chemin acceptant λ, μ étiquetés par le même mot u et pour chaque préfixe v de u, les chemins préfixes de λ et μ étiquetés par v se terminent à des sommets de même niveau : pour $G \in R^\omega$,

$$\left.\begin{array}{r} s_0 \xrightarrow[G]{a_1} s_1 \ldots \xrightarrow[G]{a_n} s_n \\ t_0 \xrightarrow[G]{a_1} \ldots t_1 \xrightarrow[G]{a_n} t_n \\ is_0, \ it_0, \ fs_n, \ ft_n \in G \end{array}\right\} \implies l^R_G(s_i) = l^R_G(t_i) \ \forall i \in [0, n]$$

La notion de non-ambiguïté par niveau est une généralisation de la notion de non-ambiguïté. En effet, toute grammaire non-ambiguë est non-ambiguë par niveau et sous la condition de déterminisme par niveau (que nous définissons), les deux propriétés sont équivalentes (*cf.* lemme 6.2.2).
Une grammaire R est *déterministe par niveau* si pour tout $G \in R^\omega$ les niveaux des sommets initiaux et ceux des buts d'arcs de même source et de même étiquette sont différents :

$$is, it \in G \vee (r \xrightarrow[G]{a} s \wedge r \xrightarrow[G]{a} t) \implies s = t \vee l_G(s) \neq l_G(t).$$

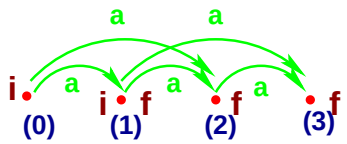


FIG. 6.4 – Graphe de grammaire déterministe par niveau.

Lemme 6.2.2. *Pour toute grammaire R déterministe par niveau, R est non-ambiguë par niveau si et seulement si R est non-ambiguë.*

Démonstration :
On doit vérifier la non-ambiguïté d'une grammaire R déterministe par niveau et non-ambiguë par niveau. Soit $G \in R^\omega$ et

$$s_0 \xrightarrow[G]{a_1} s_1 \ldots \xrightarrow[G]{a_n} s_n \wedge t_0 \xrightarrow[G]{a_1} t_1 \ldots \xrightarrow[G]{a_n} t_n \wedge is_0, it_0, fs_n, ft_n \in G.$$

Comme R est non-ambiguë par niveau, on a $l(s_i) = l(t_i)$ pour tout $i \in [0, n]$. De plus, comme R est déterministe par niveau, et par récurrence sur $i \in [0, n]$, $s_i = t_i$.

□

6.3 Synchronisation

On considère une nouvelle définition de la synchronisation. Plutôt que de synchroniser des grammaires en fonction des chemins initiaux de leurs

graphes, la relation se base sur les étiquettes et les niveaux des chemins acceptants. Ainsi, si on restreint les graphes des grammaires aux sommets co-accessibles (par f), les deux définitions sont équivalentes. Notons que pour l'étude des langages synchronisés, il suffit de considérer des grammaires restreintes aux accessibles (par i) et aux co-accessibles (par f).
Dorénavant, on dit qu'une grammaire R synchronise une grammaire S si pour (tout graphe) $G \in R^\omega$ et (tout graphe) $H \in S^\omega$, on a :

$$\begin{array}{lllll} \text{pour tout} & s_0 \xrightarrow[H]{a_1} s_1 \ldots \xrightarrow[H]{a_n} s_n & \text{avec} & is_0,\ fs_n \in H \\ \text{il existe} & r_0 \xrightarrow[G]{a_1} r_1 \ldots \xrightarrow[G]{a_n} r_n & \text{avec} & ir_0,\ fr_n \in G \\ & & \text{et} & l_G^R(r_i) = l_H^S(s_i)\ \forall i \in [0,n]. \end{array}$$

La première chose que nous vérifions est la préservation du critère de non-ambiguïté par niveau par synchronisation.

Lemme 6.3.1. *Pour toute grammaire non-ambiguë par niveau R, on a*

$$\begin{array}{rcl} S \lhd R & \Longrightarrow & S \text{ est non-ambiguë par niveau} \\ S \bowtie R & \Longleftrightarrow & S \lhd R \ \wedge\ L(S) = L(R). \end{array}$$

Démonstration :
Nous commençons par montrer la première assertion. Soit R une grammaire non-ambiguë par niveau et S telle que $S \lhd R$. Prenons $G \in R^\omega$ et $H \in S^\omega$. Considérons un mot $u = a_1 \ldots a_n$ étiquetant deux chemins dans H :

$$\begin{array}{rlll} & s_0 \xrightarrow[H]{a_1} s_1 \ldots \xrightarrow[H]{a_n} s_n & \text{avec} & is_0,\ fs_n \in H \\ \wedge & s'_0 \xrightarrow[H]{a_1} s'_1 \ldots \xrightarrow[H]{a_n} s'_n & \text{avec} & is'_0,\ fs'_n \in H. \end{array}$$

Comme $S \lhd R$, il existe des chemins étiquetés par u dans G

$$\begin{array}{rlll} & r_0 \xrightarrow[G]{a_1} r_1 \ldots \xrightarrow[G]{a_n} r_n & \text{avec} & ir_0,\ fr_n \in G \\ \wedge & r'_0 \xrightarrow[G]{a_1} r'_1 \ldots \xrightarrow[G]{a_n} r'_n & \text{avec} & ir'_0,\ fr'_n \in G \end{array}$$

tels que pour tout $1 \leq i \leq n$, $l_R^G(r_i) = l_H^S(s_i)$ et $l_R^G(r'_i) = l_H^S(s'_i)$.
Comme R est non-ambiguë par niveau, on a pour tout $i \in [0,n]$, $l_R^G(r_i) = l_R^G(r'_i)$. Ainsi, $l_H^S(s_i) = l_H^S(s'_i)$ pour tout $0 \leq i \leq n$ et donc S est non-ambiguë par niveau.
Concernant la deuxième assertion, on a par définition

$$S \bowtie R \implies S \triangleleft R \wedge L(S) = L(R).$$

Pour l'implication inverse, considérons une grammaire $S \triangleleft R$ telle que $L(S) = L(R)$. Vérifions que $R \triangleleft S$. Soient $G \in R^\omega,\ H \in S^\omega$, et $r_0 \xrightarrow[G]{a_1} r_1 \ldots \xrightarrow[G]{a_n} r_n$ avec $ir_0,\ fr_n \in G$. Le mot $a_1 \ldots a_n$ est un mot de $L(R) = L(S)$. Il existe donc un chemin $s_0 \xrightarrow[H]{a_1} s_1 \ldots \xrightarrow[H]{a_n} s_n$ avec $is_0,\ fs_n \in H$. Comme $S \triangleleft R$, il existe un chemin $r'_0 \xrightarrow[G]{a_1} r'_1 \ldots \xrightarrow[G]{a_n} r'_n$ avec $ir'_0,\ fr'_n \in G$ avec $l^R_G(r'_i) = l^S_H(s_i)$ pour tout $i \in [0, n]$. Comme R est non-ambiguë par niveau, on a $l^R_G(r_i) = l^R_G(r'_i)$ et donc $l^R_G(r_i) = l^S_H(s_i)$. Ainsi, $R \triangleleft S$.

□

On étend la propriété 4.2.6 et on montre que restreindre une grammaire par co-accessibilité à partir des sommets finaux ne modifie par la synchronisation.

Lemme 6.3.2. *Toute grammaire R peut être transformée en une grammaire S telle que $S \bowtie R \ \wedge\ S^\omega = R^\omega_{i \to f}$.*

Démonstration :
En plus de l'ensemble $Acc(A, I)$ défini dans la propriété 4.2.6, on définit, pour chaque $A \in N_R$ et $I \subseteq [\rho(A)]$, l'ensemble $CoAcc(A, I)$ des sommets de $\mathcal{S}_{H_A}$ qui sont co-accessibles par I et f dans $R^\omega(H_A)$:

$$CoAcc(A, I) := \{s \in \mathcal{S}_{H_A} \mid s \underset{R^\omega(H_A)}{\Longrightarrow} t \wedge (ft \in R^\omega(H_A) \vee t \in I\}.$$

De la même manière que pour $Acc(A, I)$, on construit les ensembles $CoAcc(A, I)$ par plus petit point fixe.
Pour chaque $A \in N_R$ et $I \subseteq [\rho(A)]$, on associe le mot

$$<A, I> := m_1 \ldots m_p \ \text{ avec } m_1 < \ldots < m_p \text{ et } \{m_1, \ldots, m_p\} = I$$

et on prend un nouveau symbole A_I d'arité $|I|$. On définit alors la grammaire

$$S := \{(A_I<A, I>, H_{A,I}) \mid A \in N_R \wedge I \subseteq [\rho(A)]\}$$

avec

$$\begin{array}{rcl} H_{A,I} & := & [H_A]_{|Acc(A,I)\cap CoAcc(A,I)} \\ & \cup & \{B_J Y(<B,J>(1))\dots Y(<B,J>(|J|)) \mid BY \in H_A \wedge B \in N_R\} \\ \text{avec} & & J = \{k \mid Y(k) \in Acc(A,I) \cap CoAcc(A,I)\} \end{array}$$

en se restreignant aux règles dont les non-terminaux sont accessibles à partir de $Z = Z_\emptyset$. Par construction, $S \bowtie R$ et $S^\omega = R^\omega_{i\to f}$.

□

La transformation d'une grammaire en une grammaire initiale (6.2.1) préserve la synchronisation, la non-ambiguïté et la non-ambiguïté par niveau des grammaires. Voici quelques propriétés des grammaires initiales obtenues par cette construction.

Proposition 6.3.3. *Pour toutes grammaires R et S, on a*
a) pour tout $G \in R^\omega$ avec $\iota, \phi \notin \mathcal{S}_G$, $G_{\iota,\phi} \cup \{i\iota, f\phi\} \in [R,\iota,\phi]^\omega$
b) $S \lhd R \iff [S,\iota,\phi] \lhd [R,\iota,\phi]$
c) $S' \lhd [R,\iota,\phi] \iff \exists S \lhd R,\ [S,\iota,\phi] \bowtie S'$
d) R est non-ambiguë par niveau $\iff$ $[R,\iota,\phi]$ est non-ambiguë par niveau
e) R est non-ambiguë $\iff$ $[R,\iota,\phi]$ est non-ambiguë.

D'après la proposition 5.3.1, la synchronisation par grammaire déterministe préserve la finitude du degré (entrant) des graphes engendrés. On étend ce résultat à la synchronisation pour toute grammaire. Pour cela, on utilise différentes propriétés sur les grammaires inverses.

Lemme 6.3.4. *Pour toutes grammaire R et S, on a*
a) $S \lhd R \iff S^{-1} \lhd R^{-1}$
b) R est non-ambiguë par niveau $\iff$ R^{-1} est non-ambiguë par niveau
c) R est non-ambiguë $\iff$ R^{-1} est non-ambiguë.

Ceci étant, il en résulte que la synchronisation préserve la finitude du degré (entrant ou sortant) des graphes engendrés.

Lemme 6.3.5. *Pour $R \rhd S$, on a*

$$\begin{array}{rcl} R^\omega \text{ est de degré entrant fini} & \Longrightarrow & S^\omega_{i\to f} \text{ est de degré entrant fini} \\ R^\omega \text{ est de degré sortant fini} & \Longrightarrow & S^\omega_{i\to f} \text{ est de degré sortant fini.} \end{array}$$

Démonstration :
Soient $G \in R^\omega$ et $H \in S^\omega_{i\to f}$.
Commençons par montrer la première assertion, par contraposée. On suppose que H est de degré entrant infini et on montre que G est aussi de degré entrant infini. Il existe donc un sommet p de H tel que

$$p_0 \xrightarrow[H]{a_0} p \ , \ \ldots \ , \ p_n \xrightarrow[H]{a_n} p \ , \ \ldots$$

avec $p_i \neq p_j$ pour tout $i \neq j$. Par définition du niveau d'un sommet, l'ensemble des sommets de même niveau est fini : $\forall n \geq 0,\ |\{q \in \mathcal{S}_H \mid l^S_H(q) = n\}| < \omega$. Donc on peut supposer que $l^S_H(p_0) \neq l^S_H(p_1) \neq \ldots \neq l^S_H(p_n) \neq \ldots$.
Par accessibilité des sommets p_n à partir de i, il existe pour tout $n \geq 0$, un sommet r_n et un mot u_n tels que $r_n \xRightarrow[H]{u_n} p_n$ et $ir_n \in H$. De même, par co-accessibilité de p à partir de f, il existe (au moins) un sommet $q \in H$ et un mot v tels que $p \xRightarrow[H]{v} q$ et $fq \in H$.
Comme $S \lhd R$, il existe pour tout $n \geq 0$

$$\begin{aligned} & s'_n \xRightarrow[G]{u_n} s_n \xrightarrow[G]{a_n} t_n \xRightarrow[G]{v} t'_n \text{ avec } is'_n, ft'_n \in G \\ \wedge \ \ & l^R_G(s_n) = l^S_H(p_n) \ \wedge \ l^R_G(t_n) = l^S_H(p). \end{aligned}$$

En particulier, on a

$$\begin{aligned} & l^R_G(s_0) \neq l^R_G(s_1) \neq \ldots \neq l^R_G(s_n) \neq \ldots \\ \text{et} \quad & l^R_G(t_0) = l^R_G(t_1) = \ldots = l^R_G(t_n) = \ldots. \end{aligned}$$

Par finitude du nombre de sommets de même poids dans G, il y a un sous-ensemble infini $I \subseteq I\!N$ tel que pour tout $m, n \in I$, $t_m = t_n$. En posant t le sommet associé à I, on a $s_n \xrightarrow[G]{a} t$ pour tout $n \in I$. Et donc t est de degré entrant infini.
Concernant la deuxième implication, considérons que R^ω est de degré sortant fini. Donc, $(R^{-1})^\omega = (R^\omega)^{-1}$ est de degré entrant fini. Par la première propriété du lemme 6.3.4, on a $R^{-1} \rhd S^{-1}$ et par la première implication, on a $(S^{-1})^\omega_{i\to f}$ est de degré entrant fini. Comme $(S^{-1})^\omega_{i\to f} = (S^\omega_{i\to f})^{-1}$, on a $S^\omega_{i\to f}$ est de degré sortant fini.

□

Par les lemmes 5.4.1 et 5.4.5, on a montré que l'ensemble des langages synchronisés par un synchronisateur R engendrant un graphe déterministe d'une part contient les langages réguliers inclus dans $L(R)$ et d'autre part, est clos pas l'union ensembliste. On étend ce résultat pour tout synchronisateur.

Proposition 6.3.6. *Pour toute grammaire R, l'ensemble $Sync(R)$ des langages synchronisés par R est clos par union et contient $L(R) \cap L$ pour tout langage régulier L.*

Démonstration :
Commençons par établir que $Sync(R)$ est fermé par union.
Soient S et S' deux grammaires synchronisées par R. On note (Z, H) (respectivement (Z, H')) la règle de l'axiome de S (respectivement S'). A renommage près, on suppose que $\mathcal{S}_H \cap \mathcal{S}_{H'} = \emptyset$ et $N_S \cap N_{S'} = \{Z\}$. On définit la grammaire

$$S + S' \ := \ \{(Z, H \cup H')\} \cup (S - \{(Z, H)\}) \cup (S' - \{(Z, H')\}).$$

Ainsi, on a $S + S' \triangleleft R$ et

$$(S + S')^\omega \ = \ \{G \cup G' \mid G \in S^\omega \wedge G' \in S'^\omega \wedge \mathcal{S}_G \cap \mathcal{S}_{G'} = \emptyset\}$$

et donc $L(S + S') = L(S) \cup L(S')$.
Il reste à établir que $Sync(R)$ contient $L(R) \cap L$ pour tout langage régulier L. Pour cela, il suffit d'appliquer la construction du lemme 5.4.1.

□

Dans les chapitres précédents, nous avons montré que pour une grammaire pondérée R, l'ensemble $Sync(R)$ des langages synchronisés par R forme une algébre de Boole (théorème 5.4.6). On étend ce résultat en considérant R non-ambiguë. Néanmoins, les opérations utilisées précédemment ne sont pas suffisantes. En effet, la non-ambiguïté est une propriété globale et non pas une propriété locale comme le sont la pondération et le déterminisme de graphes. Mais la synchronisation et la notion de non-ambiguïté par niveau permettent de se ramener à un niveau local, et plus exactement niveau par niveau.

On utilise d'autres couleurs en plus de i et f. Rappelons que pour tout $C \subseteq F_1 - \{i\}$ et toute grammaire R, on note R_C la grammaire obtenue à partir de R en remplaçant la couleur des sommets coloriés dans C par f et en enlevant f sur les autres sommets :

$$R_C := \{(X, (H - \{f\}\mathcal{S}_H) \cup \{fp \mid \exists c \in C,\ cp \in H\}) \mid (X, H) \in R\}.$$

En utilisant le produit de synchronisation par niveau, et par recoloriage des sommets finaux, on a les propriétés ci-dessous.

Lemme 6.3.7. *Pour toutes grammaires R et S on a :*

$$(R \times_l S)^\omega = R^\omega \times_l S^\omega \ ;\ (R \times_l S)_{\{f,f_1\}} \lhd R \text{ et } (R \times_l S)_{\{f,f_2\}} \lhd S \ ;$$
$$S \lhd R \implies S \bowtie R \times_l S$$

Démonstration :
Etablissons la première égalité. Soient $G \in R^\omega$ et $H \in S^\omega$. On veut montrer que $G \times_l H \in (R \times_l S)^\omega$. Il existe une dérivation

$$Z = G_0 \underset{R}{\Rightarrow} G_1 \ldots \underset{R}{\Rightarrow} G_n \ldots \text{ avec } \textstyle\bigcup_{n\geq 0}[G_n] = G$$

et une dérivation

$$Z = H_0 \underset{S}{\Rightarrow} H_1 \ldots \underset{S}{\Rightarrow} H_n \ldots \text{ avec } \textstyle\bigcup_{n\geq 0}[H_n] = H.$$

Ainsi, $G \times_l H = \bigcup_{n\geq 0}[G_n] \times_l [H_n]$. On étend le produit de synchronisation par niveau des graphes aux hypergraphes. Soit P un hypergraphe étiqueté dans $N_R \cup F_1 \cup F_2$ et une application $l_P : \mathcal{S}_P \to I\!N$ associant à chaque sommet $s \in \mathcal{S}_P$ son niveau $l_P(s)$. Soit Q un hypergraphe étiqueté dans $N_S \cup F_1 \cup F_2$ et une application $l_Q : \mathcal{S}_Q \to I\!N$ associant à chaque sommet $s \in \mathcal{S}_Q$ son niveau $l_Q(s)$. On définit le produit de synchronisation par niveau de P par Q comme suit :

$$\begin{array}{rcl} P \times_l Q & := & [P] \times_l [Q] \\ & \cup & \{[AX, BY, E] \mid AX \in P \wedge BY \in Q \wedge A \in N_R \wedge B \in N_S\} \\ \text{avec } E & = & \{(i,j) \in [\rho(A)] \times [\rho(B)] \mid l_P(X(i)) = l_Q(Y(j))\}. \end{array}$$

En particulier, $[P \times_l Q] = [P] \times_l [Q]$. Soient $P \underset{R}{\Rightarrow} P'$ et $Q \underset{S}{\Rightarrow} Q'$. Prenant

$$m := max(\{l_P(s) \mid s \in \mathcal{S}_P\} \cup \{l_Q(s) \mid s \in \mathcal{S}_Q\})$$

on définit

$$l_{P'}(s) := \begin{cases} l_P(s) & \text{si } s \in \mathcal{S}_P \\ m+1 & \text{si } s \in \mathcal{S}_{P'} - \mathcal{S}_P \end{cases} \quad \text{et} \quad l_{Q'}(s) := \begin{cases} l_Q(s) & \text{si } s \in \mathcal{S}_Q \\ m+1 & \text{si } s \in \mathcal{S}_{Q'} - \mathcal{S}_Q. \end{cases}$$

Ainsi, $P \times_l Q \underset{R\times_l S}{\Rightarrow} P' \times_l Q'$.
Par récurrence sur $n \geq 0$, on a $G_n \times_l H_n \underset{R\times_l S}{\Rightarrow} G_{n+1} \times_l H_{n+1}$ et donc $\bigcup_{n\geq 0}[G_n \times_l H_n] \in (R \times_l S)^\omega$.
Finalement, $G \times_l H = \bigcup_{n\geq 0}[G_n] \times_l [H_n] = \bigcup_{n\geq 0}[G_n \times_l H_n] \in (R \times_l S)^\omega$.

□

Le résultat de clôture de $Sync(R)$ par intersection (lemme 5.4.3) s'étend alors si R est non-ambiguë par niveau.

Lemme 6.3.8. *Pour toute grammaire R non-ambiguë par niveau et pour tout S, $S' \triangleleft R$, $L(S \times_l S') = L(S) \cap L(S')$.*

Démonstration :
Par le lemme 6.3.7, $S \times_l S' \triangleleft S$ et $S \times_l S' \triangleleft S'$, et donc $L(S \times_l S') \subseteq L(S) \cap L(S')$.
Concernant l'inclusion inverse, soit $u = a_1 \dots a_n \in L(S) \cap L(S')$. Pour $H \in S^\omega$ et $H' \in S'^\omega$, il existe

$$s_0 \xrightarrow[H]{a_1} s_1 \dots \xrightarrow[H]{a_n} s_n \wedge s'_0 \xrightarrow[H']{a_1} s'_1 \dots \xrightarrow[H']{a_n} s'_n \wedge is_0, fs_n \in H \wedge is'_0, fs'_n \in H'.$$

Comme $S, S' \triangleleft R$ et pour $G \in R^\omega$, il existe

$$r_0 \xrightarrow[G]{a_1} r_1 \dots \xrightarrow[G]{a_n} r_n \wedge r'_0 \xrightarrow[G]{a_1} r'_1 \dots \xrightarrow[G]{a_n} r'_n \wedge ir_0, ir'_0, fr_n, fr'_n \in G$$

tel que pour tout $i \in [0, n]$, $l^R_G(r_i) = l^S_H(s_i)$ et $l^R_G(r'_i) = l^S_H(s'_i)$.
Comme R est non-ambiguë par niveau, pour tout $i \in [0, n]$, $l^R_G(r_i) = l^R_G(r'_i)$ et donc $l^S_H(s_i) = l^{S'}_{H'}(s'_i)$.
Ainsi, $(s_0, s'_0) \xrightarrow[H\times_l H']{a_1} (s_1, s'_1) \dots \xrightarrow[H\times_l H']{a_n} (s_n, s'_n)$ avec $i(s_0, s'_0), f(s_n, s'_n) \in H \times_l H'$.
Et donc, $u \in L(H \times_l H')$ *i.e.* $u \in L(S \times_l S')$ par le lemme 6.3.7.

□

Dans le chapitre précédent, nous avions défini le produit de synchronisation étendu par niveau, que nous réutilisons ici pour montrer d'une part la décidabilité de la synchronisation par une grammaire déterministe par niveau et d'autre part la clôture par complémentaire.

Lemme 6.3.9. *Pour toutes grammaires R et S, si R est déterministe par niveau alors $R \triangleright S$ est décidable.*

Démonstration :
D'après le lemme 6.3.3, on suppose que R et S ont un unique sommet initial qui est situé dans l'axiome. Par la propriété 4.2.6, on construit une grammaire $(R \otimes_l S)_{i\rightarrow}$ qui engendre $(R \otimes_l S)^\omega$ restreint aux sommets accessibles à partir de i. Il reste à vérifier que

$$R \triangleright S \iff (R \otimes_l S)_{i\rightarrow} \text{ n'a pas de sommets étiquetés par } f_2 \text{ ou } f^2.$$

Soient $G \in R^\omega$ et $H \in S^\omega$.
($\Leftarrow$) : on montre par contraposée que si $(R \otimes_l S)_{i\rightarrow}$ n'a aucun sommet étiqueté par f_2 ou f^2 alors $R \triangleright S$. Supposons que R ne synchronise pas S. Il existe alors un chemin dans H tel que

$$t_0 \xrightarrow[H]{a_1} t_1 \dots \xrightarrow[H]{a_n} t_n \text{ avec } it_0, ft_n \in H$$

et qui n'est synchronisé par aucun chemin acceptant dans G.
Soit $m \leq n$ l'entier maximum tel qu'il existe

$$s_0 \xrightarrow[G]{a_1} s_1 \dots \xrightarrow[G]{a_m} s_m \text{ avec } is_0 \in G \text{ et } l^R_G(s_i) = l^S_H(t_i) \text{ pour tout } i \in [m].$$

Ainsi, $(s_0, t_0) \xrightarrow[G\otimes_l H]{a_1} (s_1, t_1) \dots \xrightarrow[G\otimes_l H]{a_m} (s_m, t_m)$ et $i(s_0, t_0) \in G \otimes_l H$.
On distingue deux cas :
a) si $m = n$.
Par hypothèse, $fs_n \notin G$ et donc $f_2(s_n, t_n) \in (G \otimes_l H)_i$. Donc $(R \otimes_l S)_{i\rightarrow}$ a un sommet étiqueté par f_2.
b) si $m < n$.

Par maximalité de m, on a $(s_m, t_m) \underset{G\otimes_l H}{\overset{a_{m+1}}{\rightarrow}} (\perp, t_{m+1}) \dots \underset{G\otimes_l H}{\overset{a_n}{\rightarrow}} (\perp, t_n)$.
Ainsi, $f^2(\perp, t_n) \in (G \otimes_l H)_i$ et donc $(R \otimes_l S)_{i\rightarrow}$ a un sommet étiqueté par f^2.

($\Rightarrow$) : on montre par contradiction que si $R \triangleright S$ alors $(R \otimes_l S)_{i\rightarrow}$ n'a pas de sommet colorié par f_2 ou f^2. Supposons que $R \triangleright S$ et qu'il y a un chemin

$$(s_0, t_0) \underset{G\otimes_l H}{\overset{a_1}{\rightarrow}} (s_1, t_1) \dots \underset{G\otimes_l H}{\overset{a_n}{\rightarrow}} (s_n, t_n)$$

avec $i(s_0, t_0) \in G \otimes_l H$ et $f_2(s_n, t_n) \in G \otimes_l H \ \vee \ f^2(s_n, t_n) \in G \otimes_l H$.
On a alors

$$t_0 \underset{H}{\overset{a_1}{\rightarrow}} t_1 \dots \underset{H}{\overset{a_n}{\rightarrow}} t_n \text{ avec } it_0, ft_n \in H$$

et pour $m = max\{i \mid s_i \neq \perp\}$,

$$s_0 \underset{G}{\overset{a_1}{\rightarrow}} s_1 \dots \underset{G}{\overset{a_m}{\rightarrow}} s_m \text{ avec } is_0 \in G \text{ et } l_G(s_i) = l_H(t_i) \text{ pour tout } i \in [m].$$

Comme $R \triangleright S$, il existe

$$r_0 \underset{G}{\overset{a_1}{\rightarrow}} r_1 \dots \underset{G}{\overset{a_n}{\rightarrow}} r_n \text{ avec } ir_0, fr_n \in G \text{ et } l_G(r_i) = l_H(t_i) \text{ pour tout } i \in [n].$$

Comme R est déterministe par niveau et par récurrence sur $i \in [m]$, on a $r_i = s_i$. On distingue les deux cas suivants :
a) : $m < n$.
On a $s_m \underset{G}{\overset{a_{m+1}}{\rightarrow}} r_{m+1}$. Par définition de $G \otimes_l H$, on a $l_G(r_{m+1}) \neq l_H(t_{m+1})$ ce qui est contradictoire.
b) : $m = n$.
On a $f_2(s_n, t_n)$ et donc $fs_n \notin G$, ce qui est contradictoire avec $r_n = s_n$.

□

La synchronisation selon les chemins acceptant permet d'étendre le lemme 5.3.4 pour toutes grammaires.

Lemme 6.3.10. *Pour toutes grammaires R et S, on a*

$$(R \otimes_l S)_{\{f,f_1,f^1\}} \bowtie R \,;\, (R \otimes_l S)_{\{f,f_2,f^2\}} \bowtie S \,;$$
$$(R \otimes_l S)_c \bowtie (R \times_l S)_c \ \forall c \in \{f, f_1, f_2\}$$

.

Le langage $L(R)-L(S)$ pour $S \triangleleft R$ est l'ensemble des mots non-acceptants étiquetant dans $(R \otimes_l S)^\omega$ un chemin initial se terminant par un sommet colorié par f_1 ou f^1 :

$$\begin{aligned} L(R) - L(S) &= L(R) - (L(R) \cap L(S)) \\ &= L((R \otimes_l S)_{\{f,f_1,f^1\}}) - L(R \otimes_l S) \\ &= L((R \otimes_l S)_{\{f_1,f^1\}}) - L(R \otimes_l S). \end{aligned}$$

Quand $(R \otimes_l S)_{\{f,f_1,f^1\}}$ est non-ambiguë, le langage

$$L((R \otimes_l S)_{\{f_1,f^1\}}) - L(R \otimes_l S)$$

est l'ensemble des mots étiquetant les chemins initiaux finissant à des sommets non-finaux coloriés par f_1 ou f^1. Plus précisément, pour f_0 une nouvelle couleur, on note par f_0-f (respectivement f_0+f) pour désigner les sommets non-finaux (resp. finaux) coloriés par f_0.

Lemme 6.3.11. *Pour toute grammaire R telle que R_{f,f_0} est non-ambiguë, on a $L(R_{f_0-f}) = L(R_{f_0}) - L(R)$ et $L(R_{f_0+f}) = L(R_{f_0}) \cap L(R)$.*

Démonstration :
Pour toute grammaire R, on a

$$L(R_{f_0}) - L(R) \subseteq L(R_{f_0-f}) \text{ et } L(R_{f_0+f}) \subseteq L(R_{f_0}) \cap L(R).$$

Soit $G \in R^\omega$. Pour tout mot $u \in L(R_{f_0-f})$, il existe un sommet p de G tel que

$$s \xRightarrow[G]{u} p \ \wedge \ is, f_0p \in G \ \wedge \ fp \notin G.$$

En particulier, $u \in L(R_{f_0})$. Comme R_{f,f_0} est non-ambiguë, on a $u \notin L(R)$. Ainsi, $L(R_{f_0-f}) \subseteq L(R_{f_0}) - L(R)$.
Concernant l'autre inclusion, soit $u \in L(R_{f_0}) \cap L(R)$, il existe

$$s \xRightarrow[G]{u} p \ \wedge \ s' \xRightarrow[G]{u} p' \ \wedge \ is, f_0p, is', fp' \in G.$$

Comme R_{f,f_0} est non-ambiguë, on a $p = p'$. Ainsi, $u \in L(R_{f_0+f})$.

□

Par les lemmes 6.3.10 et 6.3.1, $(R \otimes_l S)_{\{f,f_1,f^1\}}$ est non-ambiguë par niveau si R est non-ambiguë par niveau. Pour obtenir la clôture par complément de $Sync(R)$ pour R non-ambiguë par niveau, on transforme R en une grammaire non-ambiguë S équivalente. Plus exactement on veut que pour $R_{\{f,f_0\}}$ non-ambiguë par niveau, on ait S non-ambiguë et $L(S_{\{f\}}) = L(R_{\{f\}})$ et $L(S_{\{f_0\}}) = L(R_{\{f_0\}})$. Cette transformation permet alors de déterminiser les chemins acceptants de même étiquette. Comme pour les produits de synchronisation par niveau, on restreint la construction usuelle de déterminisation (dite du "powerset") pour qu'elle préserve les niveaux.
La *déterminisation par niveau* de toute grammaire R est

$$Det_l(R^\omega) \ := \ \{K \mid G \in R^\omega \ \wedge \ K \simeq Det_l(G)\}$$

où la déterminisation par niveau $Det_l(G)$ de tout graphe $G \in R^\omega$ est défini par

$$\begin{array}{rcl} Det_l(G) & := & \{P \xrightarrow{a} Q \mid P, Q \in \Pi \wedge Q \subseteq Succ_a(P) \wedge \\ & & \qquad \forall q \in Succ_a(P) - Q, Q \cup \{q\} \notin \Pi\} \\ & \cup & \{iP \mid P \in \Pi \wedge \forall p \in P \ ip \in G \wedge \\ & & \qquad \forall q (iq \in G \wedge q \notin P \implies P \cup \{q\} \notin \Pi)\} \\ & \cup & \{cP \mid P \in \Pi \wedge c \in F_1 - \{i\} \wedge \exists p \in P \ cp \in G\} \end{array}$$

restreint aux sommets accessibles à partir de i et tel que Π est l'ensemble des sous-ensembles des sommets de même niveau :

$$\Pi \ := \ \{P \mid \emptyset \neq P \subseteq \mathcal{S}_G \wedge \forall p, q \in P, l(p) = l(q)\}$$

et $Succ_a(P)$ est l'ensemble des successeurs des sommets de $P \in \Pi$ par $a \in F_G \cap F_2$:

$$Succ_a(P) \ := \ \{q \mid \exists p \in P (p \xrightarrow[G]{a} q)\}.$$

Contrairement au produit de synchronisation, Det_l ne préserve pas la régularité. Néanmoins, $Det_l(R^\omega)$ peut être engendré par une grammaire quand R est en forme normale que l'on peut obtenir par bi-synchronisation.
La construction présentée au lemme 5.3.7 préserve la non-ambiguïté.

Lemme 6.3.12. *Toute grammaire initiale (resp. non-ambiguë) peut être transformée en une grammaire de chemins (resp. non-ambiguë) bi-synchronisée.*

Pour toute grammaire de chemins R, $Det_l(R^\omega)$ peut être engendré par une grammaire $Det_l(R)$ que l'on peut définir.
Soit R une grammaire de chemins restreinte aux accessibles à partir de i. Pour chaque non-terminal $A \in N_R - \{Z\}$, on associe un nouveau symbole $\overline{A}$ d'arité 2 et on définit la grammaire $\overline{R}$ à partir de R en ajoutant les règles $\overline{A}12 \to H_A$ pour tout $A \in N_R - \{Z\}$, et en remplaçant dans les membres droits des règles tout arc non-terminal $s \xrightarrow{B} 2$ par $s \xrightarrow{\overline{B}} 2$:

$$\begin{array}{rcl}
\overline{R} & := & \{(Z, H_Z)\} \\
& \cup & \{(A12, (H_A - N_R\mathcal{S}_{H_A}2) \cup \{\overline{B}s2 \mid B \in N_R \wedge Bs2 \in H_A\}) \\
& & \qquad \mid A \in N_R - \{Z\}\} \\
& \cup & \{(\overline{A}12, (H_A - N_R\mathcal{S}_{H_A}2) \cup \{\overline{B}s2 \mid B \in N_R \wedge Bs2 \in H_A\}) \\
& & \qquad \mid A \in N_R - \{Z\}\}.
\end{array}$$

Soit $<$ un ordre sur $2^{N_{\overline{R}}-\{Z\}}$. Pour chaque sous-ensemble $\emptyset \neq P \subseteq N_{\overline{R}} - \{Z\}$, on associe un nouveau symbole P' d'arité $2^{|P|}$ et un hyperarc $<P> = P'p_1 \ldots p_m$ avec $\{p_1, \ldots p_m\} = 2^P$ et $p_1 < \ldots < p_m$, et on considère un graphe H_P tel que

$$\{Z \xrightarrow{A} A \mid A \in P\} \cup \{iZ\} \underset{\overline{R}}{\Rightarrow} H_P$$

et pour $P = \emptyset$, on définit $<\emptyset> = Z$ et $H_\emptyset = H_Z$.
Pour chaque $P \subseteq N_{\overline{R}} - \{Z\}$, on applique à H_P la déterminisation par niveau pour obtenir le graphe

$$H'_P := (Det_l(H_P) - \{i\{Z\}\})[\emptyset/\{Z\}]$$

où le niveau l d'un sommet est défini par

$$\begin{array}{rcll}
l(A) & = & 0 & \forall A \in P - N_R \\
l(A) & = & 1 & \forall A \in P \cap N_R \\
l(s) & = & 2 & \forall s \in \mathcal{S}_{H_P} - (P \cup \{Z\}).
\end{array}$$

Pour chaque $P \subseteq N_{\overline{R}} - \{Z\}$ on associe la règle suivante :

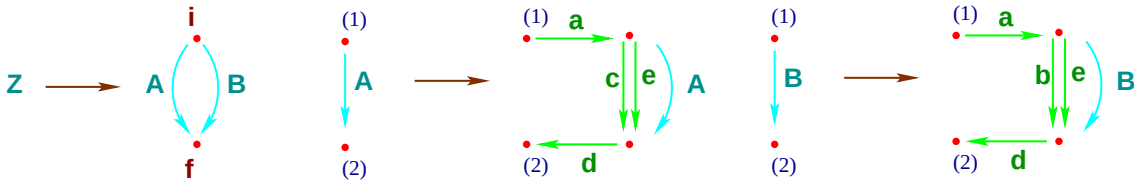
$$<P> \to [H'_P] \cup \{<Q>[U/\emptyset][U_E/E]_{\emptyset \neq E \subseteq Q} \mid U \subseteq \mathcal{S}_{H'_P} \wedge Q \neq \emptyset\}$$

avec

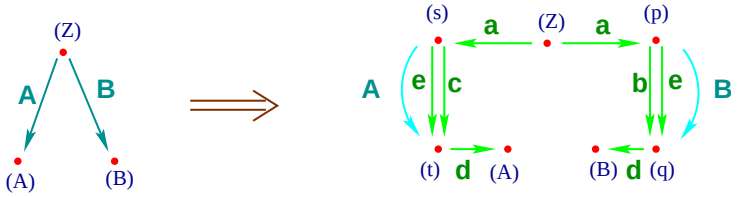
$$\begin{aligned} Q &:= \{A \in N_{\overline{R}} \mid \exists s \in U,\ s \xrightarrow[H'_P]{A}\} \\ U_E &:= \{t \mid \exists s \in U,\ \exists A \in E,\ s \xrightarrow[H'_P]{A} t\} \quad \text{pour tout } \emptyset \neq E \subseteq Q. \end{aligned}$$

En prenant toutes les règles accessibles à partir de Z, on obtient une grammaire $Det_l(R)$ engendrant $Det_l(R^\omega)$.
Remarquons que pour R non-ambiguë, on peut restreindre la définition des hyperarcs en prenant $<P> = P'p_1 \dots p_m$ avec $\{p_1, \dots, p_m\} = P$.

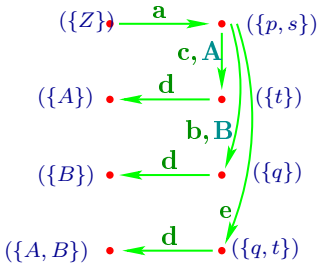
Appliquons cette construction à la grammaire R suivante :



Prenant $l(A) = l(B) = 1$ et $l(p) = l(q) = l(r) = l(s) = 2$, on a la récriture parallèle suivante :



dont le membre droit $H_{A,B}$ donne par déterminisation par niveau le graphe $Det_l(H_{A,B})$ suivant :



et la grammaire $Det_l(R)$ décrite ci-dessous

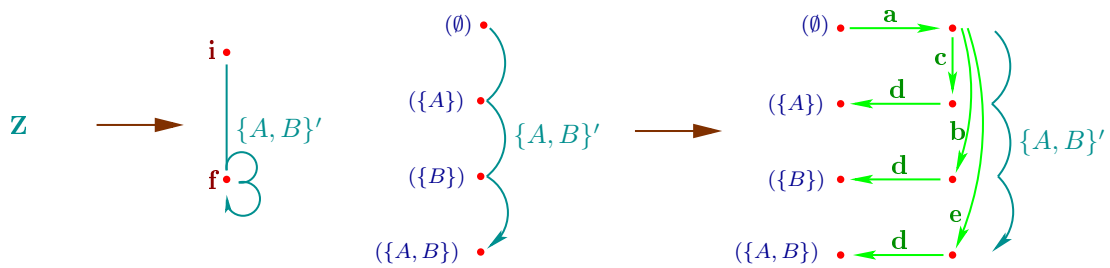
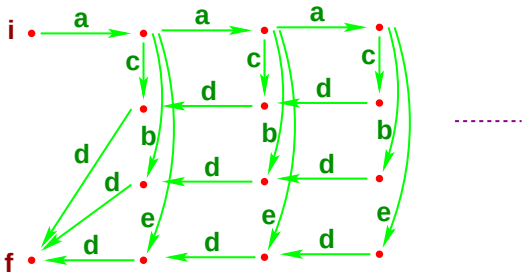


FIG. 6.5 – Determinisation par niveau.

qui engendre le graphe



Lemme 6.3.13. *Pour toute grammaire de chemins R, on a*

$$\begin{array}{rcl} (Det_l(R))^\omega & = & Det_l(R^\omega) \\ Det_l(R) & \bowtie & R \\ Det_l(R) & est & déterministe\ par\ niveau \\ R\ est\ non\text{-}ambiguë\ par\ niveau & \Longrightarrow & Det_l(R)\ est\ non\text{-}ambiguë. \end{array}$$

Par les lemmes 6.3.12 et 6.3.13, les grammaires non-ambiguës par niveau et les grammaires non-ambiguës définissent les mêmes familles de langages synchronisés.

Proposition 6.3.14. *Les grammaires (de graphes) non-ambiguës reconnaissent les langages algébriques non-ambigus.*

Démonstration :
a) Soit R une grammaire non-ambiguë. Par le lemme 6.3.3, $[R, \iota, \phi]$ est une grammaire initiale non-ambiguë reconnaissant $L([R, \iota, \phi]) = \iota L(R)\phi$. Par le

lemme 6.3.12, on transforme $[R, \iota, \phi]$ en une grammaire de chemins non-ambiguë S reconnaissant $L(S) = \iota L(R)\phi$.
Pour chaque non-terminal $A \in N_S$, on définit un renommage de sommets h_A du membre droit H_A tel que pour $A \neq Z$

$$h_A(1) = A \text{ et } h_A(2) = \varepsilon$$
$$Im(h_A) \cap Im(h_B) = \{\varepsilon\} \text{ pour tout } B \in N_S - \{A, Z\}$$
$$Im(h_A) \cap Im(h_Z) = \emptyset.$$

Soit un symbole $I \notin \bigcup\{Im(h_A) \mid A \in N_S\}$. On définit la grammaire algébrique suivante P :

$$\begin{aligned} P &:= \{(h_A(s), a h_A(t)) \mid A \in N_S \wedge s \xrightarrow[H_A]{a} t\} \\ &\cup \{(I, h_Z(s)) \mid is \in H_Z\} \\ &\cup \{(h_Z(s), \varepsilon) \mid fs \in H_Z\}. \end{aligned}$$

Le langage engendré par P à partir de l'axiome I est

$$L(P, I) = L(S) = \iota L(R)\phi.$$

On vérifie que P est non-ambiguë et, en remplaçant les terminaux ι et ϕ par ε dans les règles de P, on obtient une grammaire non-ambiguë engendrant $L(R)$.
b) Soit P une grammaire algébrique non-ambiguë d'axiome I. On peut supposer qu'il n'y a aucune occurrence de I dans les membres droits de P, et que P n'a aucune ε-règle sauf pour $I \to \varepsilon$ si $\varepsilon \in L(P, I)$.
On transforme P en la grammaire de chemins suivante :

$$\begin{aligned} R &:= \{(Z, \{I12, i1, f2\} \cup \{f1 \mid \varepsilon \in L(P, S)\})\} \\ &\cup \{(A12, H_A) \mid A \in Dom(P)\} \end{aligned}$$

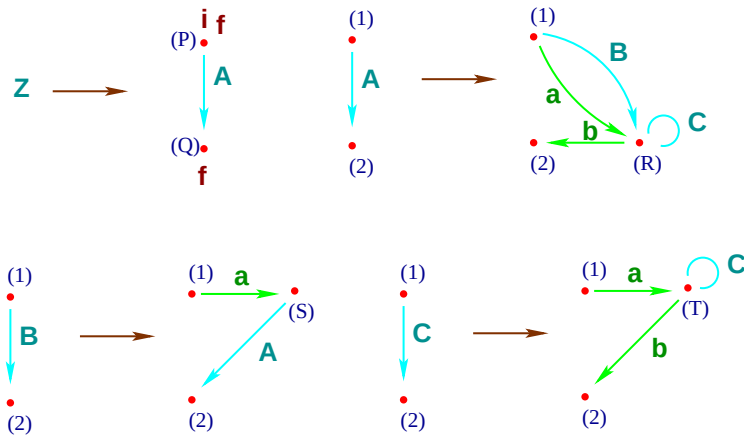
tel que pour chaque non-terminal $A \in Dom(P)$, le graphe H_A est l'ensemble des membres droits de A dans P, commençant en 1 et finissant en 2 :

$$\begin{aligned} H_A &:= \{1 \xrightarrow{B} (B, V) \mid (A, BV) \in P \wedge |B| = 1 \wedge V \neq \varepsilon\} \\ &\cup \{(U, BV) \xrightarrow{B} (UB, V) \mid (A, UBV) \in P \wedge |B| = 1 \wedge U, V \neq \varepsilon\} \\ &\cup \{(U, B) \xrightarrow{B} 2 \mid (A, UB) \in P \wedge |B| = 1 \wedge U \neq \varepsilon\} \\ &\cup \{1 \xrightarrow{B} 2 \mid (A, B) \in P \wedge |B| = 1\}. \end{aligned}$$

On a alors $L(R) = L(P, I)$ et R est une grammaire de chemins non-ambiguë.

□

Illustrons ces consructions. Tout d'abord considérons la grammaire de chemins non-ambiguë suivante :



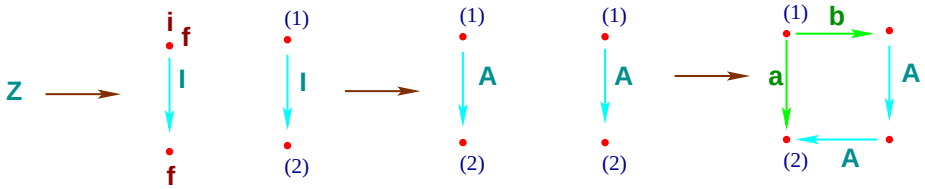
Cette grammaire accepte le même langage que la grammaire algébrique suivante, obtenue par la construction précédente.

$$\begin{array}{lllllllllll} I & = & P & ; & P & = & \varepsilon + AQ & ; & Q & = & \varepsilon \\ A & = & aR + BR & ; & R & = & b + CR & ; & B & = & aS \\ S & = & A & ; & C & = & aT & ; & T & = & b + CT. \end{array}$$

Pour illustrer la construction inverse, considérons la grammaire algébrique suivante :

$$I = \varepsilon + A \quad ; \quad A = a + bAA.$$

On la transforme en la grammaire de chemin non-ambiguë suivante :



Proposition 6.3.15. *Les grammaires non-ambiguës par niveau reconnaissent les langages algébriques non-ambigus.*

Démonstration :
Par la proposition 6.3.14, les grammaires non-ambiguës reconnaissent les langages algébriques non-ambigus. Il reste à démontrer que $L(R)$ est un langage algébrique non-ambigu pour toute grammaire R non-ambiguë par niveau. On considère deux nouveaux symboles ι et ϕ d'arité 2 et par le lemme 6.3.3, $[R, \iota, \phi]$ est une grammaire initiale non-ambiguë par niveau reconnaissant $L([R, \iota, \phi]) = \iota L(R)\phi$. Par le lemme 6.3.12, on transforme $[R, \iota, \phi]$ en une grammaire de chemins S non-ambiguë par niveau reconnaissant $L(S) = \iota L(R)\phi$. Par le lemme 6.3.13, $Det_l(S)$ est une grammaire non-ambiguë reconnaissant $L(S)$. Par la proposition 6.3.14, $\iota L(R)\phi$ est un langage algébrique non-ambigu et donc $L(R)$ l'est aussi.

□

Une autre conséquence des lemmes 6.3.12 et 6.3.13 est la clôture par complément de $Sync(R)$ quand R est non-ambiguë par niveau.

Lemme 6.3.16. *Pour R une grammaire non-ambiguë par niveau et $S \lhd R$, on a $L(R) - L(S) \in Sync(R)$.*

Démonstration :
a) Considérons que R est initiale. Donc $R \otimes_l S$ est également initiale (par rapport à $\{f, f_1, f^1\}$). Conformément au lemme 6.3.12, on construit une grammaire de chemins R' bi-synchronisée par $R \otimes_l S$ par rapport à $\{f, f_1, f^1\}$. Par le lemme 6.3.13, $Det_l(R')$ l'est également. Donc $(R \otimes_l S)_{\{f,f_1,f^1\}}$ et $R'_{\{f,f_1,f^1\}}$ sont non-ambiguës par niveau donc $(Det_l(R'))_{\{f,f_1,f^1\}}$ est non-ambiguë. Par le lemme 6.3.11,

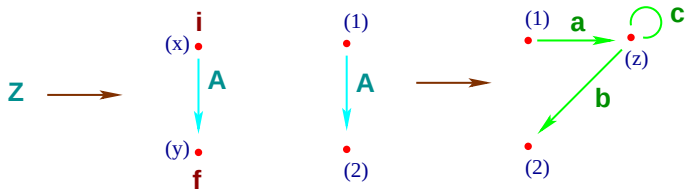
$$\begin{aligned} L((Det_l(R'))_{\{f,f_1,f^1\}}) &= L(Det_l(R')_{\{f_1,f^1\}}) - L(Det_l(R')) \\ &= L((R \otimes_l S)_{\{f_1,f^1\}}) - L(R \otimes_l S) \\ &= L(R) - L(S). \end{aligned}$$

Comme $(Det_l(R'))_{f,f_1,f^1} \bowtie R$, on a $L(R) - L(S) \in Sync(R)$.
b) Par le lemme 6.3.3, la grammaire $[R, \iota, \phi]$ est non-ambiguë par niveau telle

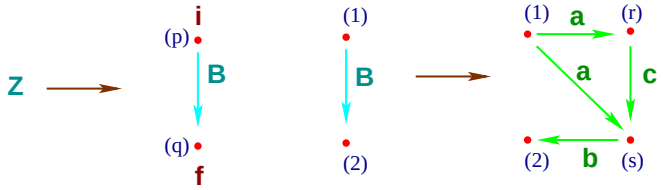
que $[S, \iota, \phi] \triangleleft [R, \iota, \phi]$. D'après (a), $L([R, \iota, \phi]) - L([S, \iota, \phi]) \in Sync([R, \iota, \phi])$.
Donc $\iota L(R)\phi - \iota L(S)\phi \in \iota Sync(R)\phi$ *i.e.* $\iota(L(R) - L(S))\phi \in \iota Sync(R)\phi$.
Donc $L(R) - L(S) \in Sync(R)$.

□

Illustrons la construction du lemme 6.3.16.
Considérons la grammaire de chemins non-ambiguë initiale R suivante :



qui reconnaît le langage $L(R) = ac^*b$.
Prenons la grammaire de chemins non-ambiguë initiale S suivante :



Donc $S \triangleleft R$ et $L(S) = \{ab, acb\}$. Appliquons la construction précédente pour construire une grammaire synchronisée par R et acceptant $L(R) - L(S) = accc^*b$. La grammaire $R \otimes_l S$ est la grammaire présentée à la figure 6.6 où

$C = [A, B, \{x, y, \perp\} \times \{p, q, \perp\} - \{(\perp, \perp)\}]$
$\xrightarrow{a}$ est une flèche d'une entrée vers une non-entrée
$\xrightarrow{b}$ est une flèche d'une non-entrée vers une entrée .
$\xrightarrow{c}$ est une flèche entre deux non-entrées.

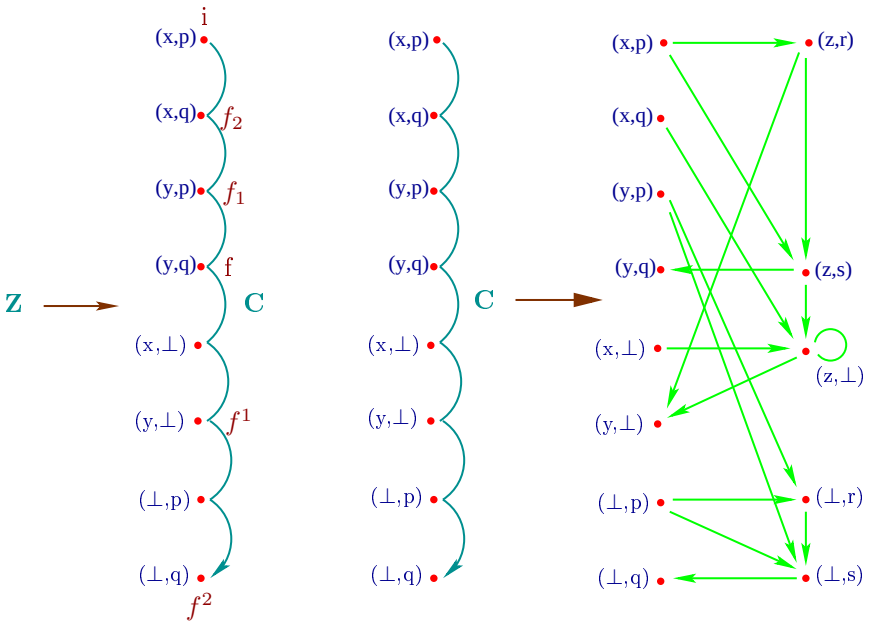
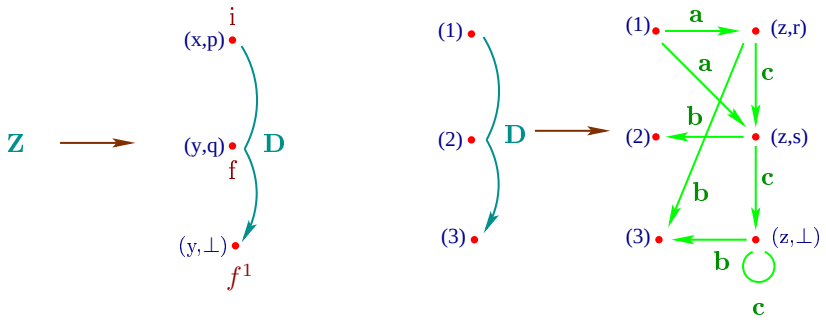


FIG. 6.6 – Produit de synchronisation par niveau de grammaires.

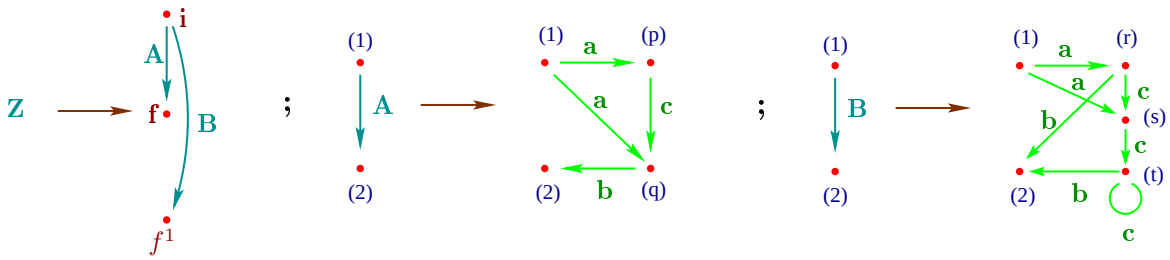
Notons que $(R \otimes_l S)_{\{f,f_1,f^1\}}$ est non-ambiguë par niveau (pour le langage $accc^*b$). Suivant le lemme 6.3.2, la restriction de $R \otimes_l S$ par accessibilité à partir de i et par co-accessibilité à partir de $\{f, f_1, f^1\}$ est la grammaire suivante :



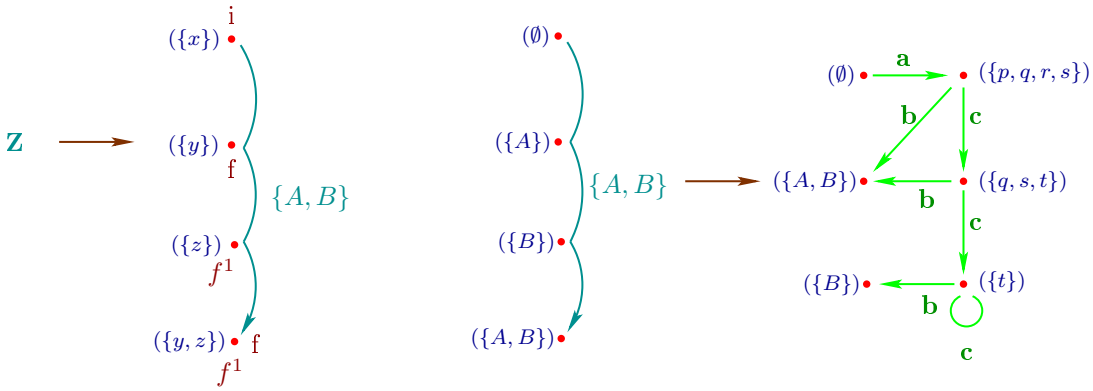
avec $D = C_{\{1,4,6\}}$ et

$$\begin{aligned} Acc(Z,\emptyset) &= CoAcc(Z,\emptyset) &&= \{(x,p),(y,q),(y,\perp)\} \\ Acc(C,\{1,4,6\}) &= CoAcc(Z,\{1,4,6\}) &&= \{1,4,6,(z,r),(z,s),(z,\perp)\}. \end{aligned}$$

Par le lemme 6.3.12, on transforme cette grammaire en la grammaire de chemin R' suivante :



Par déterminisation par niveau, on obtient la grammaire $Det_l(R')$ suivante



Ainsi, on a bien $L(R) - L(S) = accc^*b = L(Det_l(R'), i, f^1 - f) \in Sync(R)$.

Par la proposition 6.3.6 et les lemmes 6.3.8 et 6.3.16, on étend le théorème 5.4.6 pour des synchronisateurs non-ambigus par niveau.

Théorème 6.3.17. *Pour toute grammaire R non-ambiguë par niveau, la famille $Sync(R)$ forme de manière effective une algèbre booléenne (relativement à $L(R)$) de langages algébriques non-ambigus et contenant les langages réguliers inclus dans $L(R)$.*

Comme pour le cas déterministe, on peut donner une condition suffisante sur une grammaire R pour que $Sync(R)$ soit close par les opérations de concaténation et d'étoile de Kleene. Cette condition est une extension de la condition d'itérativité à des grammaires pouvant engendrer des graphes non-déterministes.

De la même manière que nous avons étendu la relation de synchronisation, *i.e.* au lieu de définir sur les chemins initiaux on définit sur les chemins acceptants, on étend également la notion d'itérativité en tenant compte des couleurs finales. Ainsi, on dit qu'une grammaire R est itérative si

- tout sommet initial est un sommet de l'axiome
- si pour tout $G \in R^\omega$ et tout mot $u = a_1 \dots a_n$:

$$r_0 \xrightarrow[G]{a_1} r_1 \dots \xrightarrow[G]{a_n} r_n \wedge ir_0, fr_n, fs \in G$$
$$\implies \exists s \xrightarrow[G]{a_1} s_1 \dots \xrightarrow[G]{a_n} s_n \text{ tel que } fs_n \in G \text{ et } l^R_G(s_i) = l^R_G(s) + l^R_G(r_i)\ \forall i \in [n]$$

Proposition 6.3.18. *Pour toute grammaire R non-ambiguë par niveau et itérative, $Sync(R)$ est fermée par concaténation et par étoile de Kleene.*

Démonstration :

i) Soit S une grammaire dont les sommets initiaux sont des sommets de son axiome K. Commençons par monter que l'on peut construire $S' \bowtie S$ tel que S'^ω n'a qu'un unique sommet initial qui n'est but d'aucun arc.

Pour chaque non-terminal $A \in N_S - \{Z\}$ et chaque $P \subseteq [\rho(A)]$, on prend un nouveau symbole A_P d'arité $\rho(A) + 1$.

On suppose que 1 n'est pas un sommet de S. L'ensemble des non-terminaux de S' est donné par

$$N_{S'} = \{Z\} \cup \{A_P \mid A \in N_S - \{Z\} \wedge P \subseteq [\rho(A)]\}.$$

L'axiome de S' est décrit par la règle

$$\begin{aligned} Z &\to [K] \cup \{i1\} \cup \{1 \xrightarrow{a} t \mid \exists s, s \xrightarrow{a} t \wedge is \in K\} \\ &\cup \{A_{\{n \mid iX(n) \in K\}} 1X \mid AX \in S \wedge A \in N_S\} \end{aligned}$$

et pour tout (AX, H) tel que $A \neq Z$ et pour tout $P \subseteq [\rho(A)]$, on prend la règle

$$\begin{aligned} A_P 1X &\to [H] \cup \{1 \xrightarrow{a} t \mid \exists p \in P,\ X(p) \xrightarrow[[H]]{a} t\} \\ &\cup \{B_{\{n \mid Y(n) \in P\}} 1Y \mid BY \in H \wedge B \in N_S\}. \end{aligned}$$

ii) Intéressons nous à la clôture par concaténation. Soit $S \triangleleft R$ et $S' \triangleleft R$ avec R itérative.

On suppose que $N_S \cap N_{S'} = \{Z\}$ et soit K l'axiome de S' *i.e.* $(Z, K) \in S'$.

Par le lemme 6.3.2, on suppose que S'^{ω} est réduit aux sommets accessibles par i et co-accessibles par f. Par (i) on peut supposer que K a un unique sommet initial r qui n'est but d'aucun arc dans S'^{ω}. Soit Z' un nouveau symbole d'arité 1.
La concaténation $S.S'$ de S par S' est la grammaire suivante :

$$\begin{array}{rcl} S.S' & := & \{(X, (H - f\mathcal{S}_H) \cup \{Z's \mid fs \in H\}) \mid (X, H) \in S\} \\ & \cup & (S' - \{(Z, K)\}) \cup \{(Z'r, K - \{ir\})\} \end{array}$$

obtenu à partir de $S \cup S'$ en remplaçant les couleurs finales de S par Z' et en ajoutant une règle qui à l'hyperarc non-terminal $Z'r$ associe le graphe K sans sommets initiaux. Comme R est itérative, $S.S' \triangleleft R$ et $L(S.S') = L(S).L(S')$.
iii) Intéressons nous à la clôture par itération de la concaténation. Soit $S \triangleleft R$. Par (i) on suppose que l'axiome K de S a un unique sommet initial r. Prenon un nouveau symbole Z' d'arité 1. L'itération S' de S est la grammaire suivante :

$$\begin{array}{rcl} S^+ & := & \{(X, H \cup \{Z's \mid fs \in H\}) \mid (X, H) \in S\} \\ & \cup & \{(Z'r, (K - \{ir\}) \cup \{Z's \mid fs \in H\})\} \end{array}$$

obtenu à partir de S en ajoutant la règle qui à partir de l'hyperarc $Z'r$ produit le graphe K sans couleur initiale et dans toutes les règles, les sommets coloriés pas f sont aussi coloriés par Z'.
Ainsi $S^+ \triangleleft R$ et $L(S^+) = (L(S))+$. Donc $(L(S))^+ \in Sync(R)$.

□

On a ainsi étendu les résultats de clôture de sous-ensembles de langages algébriques déterministes [CH08]. La synchronisation de grammaires non-ambiguës permet de définir des algèbres de Boole de langages algébriques non-ambigus qui étendent celles des langages réguliers.

Chapitre 7

Conclusion

Nous avons présenté divers travaux qui visaient à étendre des propriétés de base des langages réguliers (fermeture par concaténation, par complémentaire,...) à des sous-ensembles de langages algébriques. Les langages algébriques sont les langages acceptés par les automates à pile et c'est en travaillant sur ces objets que Mehlhorn [Mel80], Alur et Madhusudan [AM04], Caucal [Ca06] et Nowotka et Srba [NS07] ont défini des algèbres de Boole de langages algébriques déterministes étendant les langages réguliers. Nous avons étendu les automates finis aux automates réguliers acceptant les langages algébriques. Ces automates réguliers sont engendrés par les grammaires de graphes. Ces outils finis, associent à des symboles non-terminaux des graphes finis étiquetés par des symboles terminaux et non-terminaux. Partant d'un symbole non-terminal, l'application successive itérée de la grammaire, par récriture des symboles non-terminaux, engendre les graphes réguliers [Ca07]. En terme de langages, les grammaires de graphes ont la même expressivité que les automates à pile : les traces des graphes réguliers sont les langages algébriques.
Nous avons défini [CH08] une relation binaire de synchronisation entre grammaires de graphes pondérées. Les grammaires pondérées engendrent de graphes possiblement non-déterministes mais tels que le poids de tout mot initial est unique. Nous montrons que ces graphes sont déterminisables. Ainsi, les langages acceptés par les grammaires pondérées sont les langages algé-

briques déterministes. Une grammaire pondérée R synchronise une grammaire S si d'une part le langage initial accepté par S est inclus dans le langage initial accepté par R et d'autre part, les chemins initiaux de R et de S étiqueté par un même mot se terminent en des sommets de même niveau. En utilisant cette notion, nous avons montré que l'on peut définir, pour une grammaire R donnée, un sous-ensemble (strict) de langages algébriques clos par intersection et par complémentaire (par rapport au langage accepté par R). Néanmoins, dans le cas général, ces algèbres de Boole ne sont pas closes par concaténation. Mais nous donnons des conditions suffisantes sur les grammaires pour que les langages synchronisés soient clos par concaténation et son itération.
On a donc défini des sous-ensembles de langages algébriques déterministes étendant les langages réguliers : ils sont clos par intersection, complémentaire, concaténation et étoile de Kleene.
L'argument essentiel que l'on peut dégager des travaux sur les grammaires pondérées est l'unicité du niveau d'un mot. Considérant cela, Caucal [Ca08] a étudié les propriétés de synchronisation de grammaires engendrant des graphes non-ambigus. Un graphe est non-ambigu si tout mot accepté étiquette un unique chemin acceptant. En caractérisant les grammaires non-ambiguës par niveau, une généralisation des grammaires de graphes non-ambiguës, Caucal montre que la synchronisation par une grammaire non-ambiguë par niveau donnée définit une algèbre booléenne de langages non-ambigus, permettant ainsi de faire sortir les propriétés de synchronisation de la classe des langages algébriques déterministes.
Dans [Ca08], Caucal prouve également la réponse affirmative à une question soulevée lors des travaux de [CH08] : deux grammaires engendrant des graphes isomorphes définissent les mêmes ensembles de langages synchronisés.

Théorème 7.0.19 ([Ca08]). *Pour toutes grammaires R et S,*

$$R^\omega = S^\omega \implies Sync(R) = Sync(S).$$

Le théorème 7.0.19 permet ainsi de faire passer les considérations de synchronisation de grammaires à des problèmes de synchronisation de graphes,

indépendamment de la manière dont ceux-ci sont engendrés. Ceci implique que l'on puisse définir une relation de synchronisation qui ne tienne pas compte du niveau des sommets des graphes (ou du niveau des mots du langage accepté par un graphe). Les propriétés des langages (ou des graphes) qui permettraient d'étendre la synchronisation à un niveau purement structurel sont encore à découvrir.

Une autre propriété à exhiber concerne la clôture par concaténation. Dans les travaux présentés ici, la classe des langages synchronisés par une grammaire est close par concaténation si cette grammaire est itérative. Mais cette condition n'est pas *sine qua non*. On peut donc se poser la question de l'existence d'une condition nécessaire et suffisante pour qu'une classe de langages synchronisés par une grammaire soit close par concaténation.

D'autres extensions à ces travaux pourraient voir le jour en considérant d'autres classes de langages. Une extension des travaux de [AM04] a été effectuée par La Torre, Madhusudan et Parlato. Dans [LMP07], les auteurs s'intéressent aux langages des automates « visibles » à deux piles. Ils montrent alors que ces langages contextuels sont clos par les opérations booléennes et que des problèmes tels que l'inclusion et le vide d'un langage sont décidables.

La mise en œuvre de la synchronisation de grammaires a permis par ailleurs de caractériser des classes connues de langages algébriques. A titre d'exemple, nous avons donné des grammaires synchronisant exactement les langages visibles ([AM04]) ou encore les langages équilibrés ([BB02]). Néanmoins, cette mise en œuvre n'est pas si évidente et il reste à caractériser par synchronisation de grammaires les classes booléennes connues de langages algébriques.

Bibliographie

[AM04] Alur, R., Madhusudan, P. : Visibly pushdown languages. 36^{th} STOC (2004) 202–211

[AN00] Asveld, P., Nijholt, A. : The inclusion problem for some subclasses of context-free languages. TCS **230** 1-2 (2000) 247–256

[BB90] Berstel, J., Boasson, L. : Context-free languages. Chap. 2 in [Vl90] 61–102

[BB02] Berstel, J., Boasson, L. : Balanced grammars and their languages. Formal and Natural Computing, LNCS **2300** (2002) 3–25

[BE+00] Bouajjani, A., Esparza, J., Finkel, A., Maler, O., Rossmanith, P., Willems, B., Wolper, P. : An efficient automata approach to some problems on context-free grammars. Information Processing Letters, 74(5-6) (2000) 221–227

[Ca06] Caucal, D. : Synchronization of pushdown automata. 10^{th} DLT, LNCS **4036**, Ibarra O., Dang Z. (Eds.), (2006) 120-132

[Ca07] Caucal, D. : Deterministic graph grammars. Texts in Logic and Games 2(2007) 169–250

[Ca08] Caucal, D. : Boolean algebras of unambiguous context-free languages. 28^{th} FSTTCS, Dagstuhl Research Online Publication Server (2008)

[CH08] Caucal, D., Hassen, S. : Synchronisation of grammars. 3^{rd} CSR, LNCS **5010** (2008) 110–121

[Ch56] Chomsky, N. : Three models for the description of languages. IRE transactions on information theory, 2(3) (1956) 113–124

[Co89] Courcelle, B. : Infinite graphs of bounded tree width. Mathematical Systems Theory **21**(4) (1989) 187–221

[Co90] Courcelle, B. : Graph Rewriting : An Algebraic and Logic Approach. Chap. 5 in [Vl90] 193–242

[Co95] Courcelle, B. : Structural properties of context-free sets of graphs generated by vertex replacements. Information and Computation **116** (1995) 275–293

[Fr76] Friedman, E. : The inclusion problem for simple languages. TCS **1** (1976) 297–316

[GG66] Ginsburg, S., Greibach, S. : Deterministic context-free languages. Information and Control **9** (1966) 620–648

[Ha78] Harrison, M. : Introduction to formal language theory. Addison Wesley, 1978

[KH66] Korenjak, A., Hopcroft, J. : Simple Deterministic Languages. FOCS **7** (1966) 36–46

[HU69] Hopcroft, J., Ullman, J. : Formal languages and their relation to automata. Addison Wesley, 1969

[HU79] Hopcroft, J., Ullman, J. : Introduction to automata theory, languages, and computation. Addison Wesley, 1979

[Kl56] Kleene, S. : Representation of events in nerve nets and finite automata. Annals of Mathematics Studies **34** (1956) 3–41

[LMP07] La Torre, S., Madhusudan, P., Parlato, G. : A robust class of context-sensitive languages. 22^{nd} LICS (2007) 161–170

[MS97] Mateescu, A., Salomaa, A. : Aspects of classical language theory. Handbook of formal languages, Vol. 1, Springer-Verlag (1997) 175–252

[MP43] McCulloch, W., Pitts, W. : A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematicals Biophisics, 5 (1943) 115–133

[Mc67] McNaughton, R. : Parenthesis grammars. Journal of the ACM **14**(3) (1967) 490–500

[Mei07] Meitus, V. : Equivalence of deterministic pushdown automata revisited (english version). Cybernetics and System Analysis **43** (2007) 179–191

[Mel80] Mehlhorn, K. : Pebbling mountain ranges and its application to DCFL recognition. 7^{th} ICALP, LNCS **85** (1980) 422–432

[MS85] Muller, D., Schupp, P. : The theory of ends, pushdown automata, and second-order logic. Theoretical Computer Science **37** (1985) 51–75

[NS07] Nowotka, D., Srba, J. : Height-deterministic pushdown automata. 32^{nd} MFCS, LNCS **4708** (2007) 125–134

[OIH80] Oyamaguchi, M., Honda, N. : The equivalence problem for real-time strict deterministic languages. Information and Control **45** (1980) 90–115

[Pa00] Payet, E. : Produit synchronisé pour quelques classes de graphes infinis. Thèse, Université de la Réunion (2000)

[Pe90] Perrin, D. : Finite automata. Chap. 1 in [Vl90] 3–57

[Pe95] Perrin, D. : Les débuts de la théorie des automates. Technique et Science Informatique **14** (1995) 409-433

[RS59] Rabin, M., Scott, D. : Finite automata and their decision problems. IBM Journal 3 (1959) 114–125

[Ro86] Romanovskii, V. : The equivalence problem for real-time deterministic pushdown automata. Kibernetika 2 (1986) 12–23

[Sa90] Salomaa, A. : Formal languages and power series. Chap 3 in [Vl90] 105–132

[Sc55] Schützenberger, M. : Une théorie algébrique du codage. In Séminaire Dubreuil-Pisot (1955)

[Se85] Sénizergues, G. : The equivalence and inclusion problems for NTS languages. JCSS **31** (1985) 303–331

[Th97] Thomas, W. : Languages, automata, and logic. Handbook of Formal Languages, Vol. 3, Springer (1997) 389–456

[Th01] Thomas, W. : A short introduction to infinite automata. 5^{th} DLT, LNCS **2295** (2001) 130–144

[Tu36] Turing, A. : On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, Series 2, **42** (1936) 230–265
A correction, vol 43 544–546

[Va74] Valiant, L. : The decidability of equivalence for deterministic finite-turn pushdown automata. 6^{th} STOC (1974) 27–32

[VP75] Valiant, L., Patterson, M. : Deterministic one-counter automata. Journal of Computer and System Sciences **10** (1975) 340–350

[Vl90] Van Leeuwen, J. : Handbook of Theorical Computer Science, Vol. B : Formal Models and Semantics, Elsevier (1990)

Table des figures

Résumé : Les langages réguliers sont des langages qui ont été largement étudiés, notamment du point de vue de leurs propriétés de clôture ensembliste : l'ensemble des langages réguliers (pour un alphabet donné) forme une algèbre de Boole close par concaténation et étoile de Kleene. Ces propriétés ne se généralisent pas toutes à l'ensemble des langages algébriques qui est un sur-ensemble de l'ensemble des langages réguliers. Notamment les langages algébriques ne sont pas clos par intersection. Pour engendrer ces langages, nous utilisons les grammaires déterministes de graphes. Une grammaire de graphes est un système fini de récriture d'hypergraphes finis. Par récriture itérée à partir d'un non-terminal, la grammaire engendre un graphe régulier dont les traces forment un langage algébrique. En définissant une relation de synchronisation entre ces grammaires, on montre que l'on peut définir des sous-ensembles stricts de langages algébriques non-ambigus qui forment des algèbres de Boole effectives contenant les langages réguliers. Nous donnons également des conditions suffisantes pour que ces algèbres booléennes soient closes par concaténation et étoile de Kleene.

Mots-clés : algèbres de Boole, langages algébriques non-ambigus, grammaires de graphes, synchronisation de grammaires, graphes réguliers

Abstract : The regular languages have been studied for quite a long time, specially from their closure point of view : the set of regular languages (for a given alphabet) is a boolean algebra which is also closed by concatenation and the Kleene star operation. These properties do not generalize to the set of context-free languages which strictly contains the regalar languages. One can cite the fact that the context-free langages are not closed by intersection. To generate these languages, we use the deterministic graph grammars. A graph grammar is a finite set of rules defining a finite hypergraphs rewrite relation. By iterative application of this relation, we build a regular graph whose traces are a context-free language. By definition of a binary relation between grammars, the synchronisation relation, we show that one can define strict subsets of non-ambiguous context-free languages forming effective boolean algebras containing the regular languages. We also give sufficient conditions for these algebras to be closed by concatenation and the Kleene star operation.

Key words : boolean algebras, non-ambiguous context-free languages, graph grammars, synchronization of grammars, regular graphs