Chapitre 2. Langages algébriques

2.0. Grammaires formelles

Une **grammaire formelle** correspond à un tuple (A, V, P, S) avec A alphabet fini de **symboles terminaux**, V alphabet fini disjoint de A de **symboles variables**, S un élément de V appelé **axiome**, P un ensemble fini de **règles de production** sous la forme de couples notés $\boldsymbol{x \to y}$ avec $x \in (A+V)^* V (A+V)^*$ et $y \in (A+V)^*$. Autrement dit, pour une règle de production de grammaire formelle, le membre gauche est une chaine partielle avec au moins un symbole variable, et le membre droite est un chaine partielle.

L'alphabet d'une grammaire est $A + V$. **L'alphabet terminal d'une grammaire** est A.

Une **chaine (partielle) d'une grammaire** est un mot sur $A + V$.

Une **chaine terminale d'une grammaire** est un mot sur A.

Pour compacter l'écriture des productions d'une grammaire ayant même membre de gauche, par ex : $x \to y_1, x \to y_2, x \to y_3$ on écrit par abus ces 3 règles sous la forme d'une seule $x \to y_1 + y_2 + y_3$

Pour compacter l'écriture d'une grammaire on se contente souvent de donner seulement les règles de productions, l'alphabet terminal étant supposé celui des minuscules utilisées, l'alphabet de variables celui des majuscules.

Par exemple $A = \{a, b\}, V = \{S\}, P = \{S \to aSb, S \to \varepsilon\}$ peut s'écrire en mode compact : $\{S \to aSb + \varepsilon\}$

Dérivation. Sur une grammaire, une chaine partielle u **se dérive en/produit** une chaine partielle v et on note encore $\boldsymbol{u \to v}$ ssi il existe deux chaines partielles α, β et une production $x \to y \in P$ tels que $u = \alpha x \beta$ et $v = \alpha y \beta$.

Autrement dit, appliquer une règle de production consiste à remplacer dans une chaine une occurrence du membre de gauche de cette règle par son membre de droite.

La **dérivation est dite gauche (resp. droite)** ssi $\alpha \in A^*$ (resp. $\beta \in A^*$) càd ssi c'est la variable la plus à gauche (resp. droite) qui est réécrite.

Clôture des dérivations. Sur une grammaire, pour deux chaines u, v on note $\boldsymbol{u \to^1 v}$ pour $u \to v$, on note $\boldsymbol{u \to^0 v}$ pour $u = v$, on note $\boldsymbol{u \to^k v}$ pour dire $\exists u_1, \dots, u_{k-1} \in (A+V)^*$ tels que $u \to u_1 \to \cdots \to u_{k-1} \to v$, on note $\boldsymbol{u \to^* v}$ pour dire $\exists k \in N \ \ u \to^k v$.

Autrement dit, la relation $\to^*$ est la clôture réflexive transitive de la relation de dérivation $\to$.

L'application successive de règles de production $\to^*$ s'appelle encore souvent par abus une dérivation.

Le **langage partiel engendré par une chaine partielle $\boldsymbol{u}$ et une grammaire $\boldsymbol{G}$** est $\boldsymbol{\widehat{L_G}(u)} = \{v \in (A+V)^* | u \to^* v\}$

Le **langage (terminal) engendré par une chaine partielle $\boldsymbol{u}$ et une grammaire $\boldsymbol{G}$** est $\boldsymbol{L_G(u)} = \{v \in A^* | u \to^* v\} = \widehat{L_G}(u) \cap A^*$

Dans le cas où on ne précise pas, **le langage engendré par une grammaire $\boldsymbol{L_G}$** ou $\boldsymbol{\widehat{L_G}}$ est celui engendré par l'axiome S.

Un **mot partiel engendré d'une grammaire** (G, S) est un mot de $\widehat{L_G}$, càd $w \in (A+V)^* \mid S \to^* w$.

Un **mot (terminal) engendré d'une grammaire** (G, S) est un mot de L_G, càd $w \in A^* \mid S \to^* w$.

La notation L_G s'étend aux langages en posant $\boldsymbol{L_G(K)} = \bigcup_{u \in K} L_G(u)$ pour $K \subseteq (A+V)^*$.

La clôture des langages algébriques peut permettre de montrer que $L_G(K)$ reste algebrique quand K est algébrique.

2.1. Grammaires algébriques

2.1.1. Définitions et exemples

Une **grammaire algébrique (context-free)** correspond à un tuple (A, V, S, P) avec A alphabet fini de **symboles terminaux**, V alphabet fini disjoint de A de **symboles variables**, S un élément de V appelé **axiome**, et P un ensemble fini de **règles de production/dérivation** sous la forme de couples notés $\boldsymbol{X \to y}$ avec $X \in V$ et $y \in (A+V)^*$.

Une grammaire algébrique est donc une grammaire formelle dans laquelle le membre de gauche d'une production est restreint à un unique symbole variable.

Le qualificatif algébrique sera justifié parce que les langages engendrés par ce type de grammaire sont

solutions de systèmes d'équations polynomiales.
La grammaire $S \to aSb + \varepsilon$ est algébrique et engendre le langage $\{a^n b^n : n \in N\}$.
La grammaire $S \to aS + b$ est algébrique et engendre le langage $a^* b$
La grammaire $A = \{a, b, c\}, S \to AS + \varepsilon, A \to AB + a, B \to BC + b, C \to CA + c$ est algébrique.
Un **langage algébrique** est un langage engendré par une grammaire algébrique (via un axiome $S \in V$).
Exemples : Il sera montré que tout langage rationnel est algébrique.
La grammaire de Dyck d'ordre n : $A_n = \{a_1, \dots, a_n, \overline{a_1}, \dots, \overline{a_n}\}, S \to ST + \varepsilon, T \to a_1 S \overline{a_1} + \cdots + a_n S \overline{a_n}$ est algébrique. Ou plus simplement par $S \to a_i S \overline{a}_i S + \varepsilon$
Le langage de Dyck primitif d'ordre n : $D_n = L_G(T)$ est algébrique
Le langage de Dyck d'ordre n : $D_n^* = L_G(S)$ est algébrique
Si chaque lettre a_i est vue comme parenthèse ouvrante, et $\overline{a}_i$ comme la parenthèse fermante correspondante, les mots du langages de Dyck sont les mots bien parenthésés, autrement dit ce sont les mots qui se réduisent au mot vide en appliquant des réduction de la forme $a_i \overline{a}_i \to \varepsilon$
Le langage de Luckasiewicz est celui engendré par la grammaire $S \to aSS + \bar{a}$ et l'axiome S.
Le langage de Goldstine est $L = \left\{a^{n_0} b a^{n_1} b \dots a^{n_k} b : k \in N \text{ et } \exists j \in N, n_j \neq j\right\}$ et est engendré par la grammaire $T_0 \to aT_0 + \varepsilon, T_1 \to T_0 b, T_2 \to T_1 T_2 + \varepsilon, T_3 \to T_1 T_3 a + T_1 T_2 + aT_0, S \to T_3 b T_2$ et l'axiome S.
L'ensemble des palindromes sur un alphabet fini, est un langage algébrique. Son complémentaire aussi.

Sur un alphabet A, $\{ww' : w, w' \in A^*, w \neq w', |w| = |w'|\}$ est un langage algébrique.
Sur un alphabet A ne comportant pas #, $\{w\#w' : w, w' \in A^*, w \neq w'\}$ est un langage algébrique.
Lemme fondamental. Soit u, v deux chaines (partielles) d'une grammaire algébrique G, on suppose $u = u_1 u_2$. Alors $\exists k \in N\ u \to^k v$ ssi $\exists v_1, v_2$ chaines partielles, $\exists k_1, k_2 \in N, u_1 \to^{k_1} v_1,\ u_2 \to^{k_2} v_2, v = v_1 v_2, k = k_1 + k_2$.
On peut voir le sens direct par récurrence sur k ou de façon algorithmique. Sens indirect plus facile.
Lemme fondamental v2. Pour u, v deux chaines d'une grammaire algébrique G, $\widehat{L_G}(uv) = \widehat{L_G}(u)\widehat{L_G}(v)$
Tout langage rationnel inclus dans le langage engendré par $S \to aSSb + c$, est un langage fini.

2.1.2. Grammaires réduites

Une **grammaire algébrique est réduite en l'axiome $S_0 \in V$** ssi 1. et 2.
1. $\forall S \in V\ L_G(S) \neq \emptyset$, càd toute variable peut engendrer un mot terminal.
2. $\forall S \in V\ \exists u, v$ chaines tels que $S_0 \to^* uSv$, càd toute variable peut être générée en partant de S_0.
Réduction. Une grammaire algébrique muni d'un axiome (G, S_0), peut être réduite : Il existe une grammaire algébrique (G', S_0) réduite qui génère le même langage $L_G(S_0) = L_{G'}(S_0)$
L'idée de la preuve est d'abord de supprimer les variables ne pouvant produire un mot terminal, et les règles les contenant, puis de supprimer toutes les variables inaccessibles à partir de S_0, (et leur règles).

2.1.3. Grammaires propres

Une **grammaire algébrique est propre** lorsqu'aucun membre droit d'une règle n'est égal au mot vide ou à une variable.
Une grammaire algébrique G muni d'un axiome $S_0 \in V$ peut etre rendue propre : Il existe une grammaire algébrique propre G' d'axiome $S_0' \in V$ tel que $L_{G'}(S_0') = L_G(S_0) \setminus \{\varepsilon\}$.
Idée : On supprime les epsilon productions, puis on supprime les règles unitaires. Mais Il faut faire attention à ne pas changer le langage, donc les suppressions se traduisent par modification des règles.
Une **substitution de A^* vers B^*** correspond à un morphisme de monoïdes $\sigma : (A^*, \cdot) \to (P(B^*), \cdot)$
Il existe un algorithme qui détermine pour une grammaire algébrique si son langage engendré est infini ou non.

2.1.4. Forme normale quadratique (de Chomsky)

Une **grammaire algébrique est en forme normale quadratique/de Chomsky** ssi toutes ses règles sont d'une des formes : $S \to S_1 S_2$ avec $S_1 \in V, S_2 \in V$, ou $S \to A$ avec $a \in A$.

Une grammaire algébrique G muni d'un axiome $S_0 \in V$ peut etre mise en forme normale de Chomsky (G', S_0') avec $L_{G'}(S_0') = L_G(S_0) \setminus \{\varepsilon\}$. Cela peut beaucoup augmenter la taille de la grammaire.
Conséquence : Il y a un algorithme qui détermine si un mot donné est engendré par une grammaire algébrique.
Dans une grammaire en forme normale de Chomsky, chaque dérivation remplace une variable par une lettre terminale ou augmente le nombre d'occurrences de variables dans le mot. Il s'ensuit qu'une dérivation d'une variable a un mot w formé de lettres terminales est de longueur au plus $2|w|$. Pour savoir si une grammaire algébrique G engendre un mot w non vide, on remplace d'abord la grammaire G par une grammaire equivalente G' en forme normale quadratique puis on examine toutes les dérivations de longueur au plus $2|w|$ de G'. Cet algorithme n'est pas très efficace car nb de dérivations exponentiel.
L'algorithme de Cocke, Kasami, Younger résout ce problème en temps polynomial.

2.2. Système d'équations

2.2.1. Substitutions

A une grammaire algébrique on peut associer un système d'équations en langages. Par exemple, à la grammaire algébrique $\{X_1 \to X_1X_2 + \varepsilon, X_2 \to aX_2 + b\}$, on associe le système d'équations : $\begin{cases} L_1 = L_1L_2 + \varepsilon \\ L_2 = aL_2 + b \end{cases}$
On formalise cela par : Pour une grammaire algébrique $G = (A, V, P)$ avec n variables $V = \{(X_i)_{1\leq i\leq n}\}$ et n langages terminaux correspondants $L = (L_i)_{1\leq i\leq n}$ sur A^*, on définit une substitution sur l'alphabet de G par récurrence : $\boldsymbol{s_L} = \boldsymbol{s}_{(\boldsymbol{L_i})_i} : (A+V)^* \to P((A+V)^*)$:
$s_L(\varepsilon) = \{\varepsilon\},\ \forall a \in A\ s_L(a) = \{a\},\ \forall i\ s_L(X_i) = L_i,\ \forall x, y \in (A+V)^*\ s_L(xy) = s_L(x)s_L(y)$,
$\forall K \subseteq (A+V)^*\ s_L(K) = \bigcup_{w\in K} s_L(w)$. On fait bien à une substitution des X_i par les L_i correspondants.
Plutôt que d'écrire $s_L(w)$ on écrit par ex : $\boldsymbol{w(L) = w(X_1 \coloneqq L_1, \dots, X_n \coloneqq L_n)}$, car on substitue dans w.

2.2.2. Système d'équations associé à une grammaire

Le système d'équations associé à une grammaire algébrique $G = (A, V, P)$ de variables $V = \{X_1, \dots, X_n\}$ est le système $\boldsymbol{S(G)}$ de variables $(L_i)_i$ formé des n équations : $L_i = \sum_{X_i \to w} w(X_1 \coloneqq L_1, \dots, X_n \coloneqq L_n)$ pour $i \in \{1, \dots, n\}$.
Une **solution $\boldsymbol{L = (L_1, \dots, L_n})$ d'un système associé $\boldsymbol{S(G)}$** est un n-uplet de langages terminaux (sur A^*), vérifiant les équations de $S(G)$
On note parfois $L_G = \big(L_G(X_1), \dots, L_G(X_n)\big)$, mais attention car risque de confusion avec $L_G = L_G(S)$.
Reformulation lemme fondamental. Pour toute chaine partielle w d'un grammaire algébrique G de variables $\{X_1, \dots, X_n\}$, $L_G(w) = w\big(X_1 \coloneqq L_G(X_1), \dots, X_n \coloneqq L_G(X_n)\big)$

2.2.3. Existence d'une solution pour $\boldsymbol{S(G)}$

La relation essentielle entre grammaire algébrique et son système associé est que les langages engendrés forment une solution du système. Cette solution est minimale :
Minimalité langages engendrés. Pour une grammaire algébrique G de variables $\{X_1, \dots, X_n\}$, $\big(L_G(X_1), \dots, L_G(X_n)\big)$ est la solution minimale (pour l'inclusion composante par composante) de $S(G)$.

2.2.4. Unicité des solutions propres

Pour avoir unicité d'une solution, il faut se restreindre aux grammaires propres et aux solutions propres.
Un **langage $\boldsymbol{L}$ est propre** ssi il ne contient pas le mot vide ssi $\varepsilon \notin L$
Une **solution $\boldsymbol{L = (L_1, \dots, L_n})$ d'un système associé $\boldsymbol{S(G)}$ est propre** ssi tous les L_i le sont càd $\forall i\ \varepsilon \notin L_i$
Pour une grammaire algébrique propre de variables X_i, les langages engendrés par ces variables $\big(L_G(X_1), \dots, L_G(X_n)\big)$ forment l'unique solution propre du système associé à la grammaire $S(G)$.

2.2.5. Théorème de Parikh

Deux **mots sont anagrammes** ssi l'un s'obtient par une permutation des lettres de l'autre.
L'ensemble des anagrammes d'un mot $\boldsymbol{w}$ est noté $\overline{\boldsymbol{w}}$, c'est la classe d'équivalence de w.

L'image commutative d'un langage $\boldsymbol{L}$, est l'ensemble $\overline{\boldsymbol{L}} = \{\overline{w} : w \in L\}$ càd le langage quotienté par la relation d'équivalence « être anagramme ».
Deux langages sont commutativement équivalents ssi leur image commutative est identique.
Pour deux langages L, M on a toujours $\overline{L+M} = \overline{L} + \overline{M}$ et $\overline{LM} = \overline{ML}$
Parikh*. Tout langage algébrique L est commutativement équivalent à un langage rationnel R. ($\overline{L} = \overline{R}$)
Exemple : le langage $\{a^n b^n : n \in N\}$ est commutativement équivalent au langage rationnel $(ab)^*$.

2.2.6. Systèmes d'équations en commutatifs

Pour 2 chaines u, v anagrammes d'une grammaire algébrique G, les langages $u(X_1 \coloneqq L_1, \dots, X_n \coloneqq L_n)$ et $v(X_1 \coloneqq L_1, \dots, X_n \coloneqq L_n)$ sont commutativement équivalents, pour tous $(L_1, \dots, L_n)$ langages sur A.
Plus généralement, pour deux langages J et K sur $A + V$ commutativement équivalents, alors $J(X_1 \coloneqq L_1, \dots, X_n \coloneqq L_n)$ et $K(X_1 \coloneqq L_1, \dots, X_n \coloneqq L_n)$ sont encore commutativement équivalents, pour tous $(L_1, \dots, L_n)$ langages fixés sur A.
Le système commutatif d'équations associé à une grammaire algébrique $G = (A, V, P)$ de variables $V = \{X_1, \dots, X_n\}$ est le système $\overline{\boldsymbol{S(G)}}$ de variables $(L_i)_i$ formé des n équations :
$\overline{L_i} = \sum_{X_i \to w} \overline{w(X_1 \coloneqq L_1, \dots, X_n \coloneqq L_n)}$ pour $i \in \{1, \dots, n\}$.
Une **solution $\boldsymbol{L = (L_1, \dots, L_n})$ d'un système commutatif $\overline{\boldsymbol{S(G)}}$** est un n-uplet de langages terminaux (sur A^*), vérifiant les équations de $\overline{S(G)}$
Pour une grammaire algébrique propre de variables X_i, les langages engendrés par ces variables $\left(L_G(X_1), \dots, L_G(X_n)\right)$ forment une solution propre du système commutatif $\overline{S(G)}$ unique à équivalence commutative près.

2.2.7. Solutions rationnelles des systèmes commutatifs

On définit une **grammaire algébrique étendue** où la seule différence est que l'ensemble des productions de chaque variable n'est pas un ensemble fini mais un ensemble rationnel de mots : $\forall X \in V \ \{w \in (A+V)^* | X \to w \in P\}$ rationnel. Les notions de dérivation et mot engendré se généralisent. Le langage engendré est algébrique : car il suffit de rajouter des variables (autant que d'états) qui simulent chacun des automates acceptant les langages rationnels $\{w | X \to w \in P\}$.
Le système commutatif d'une grammaire algébrique étendue admet une **solution rationnelle** càd formée de n langages terminaux rationnels. Cela permet de prouver le théorème de Parikh.

2.3. Arbres de dérivation

Un **arbre de dérivation/arbre syntaxique/parse tree, sur une grammaire algébrique $\boldsymbol{G}$** correspond à un graphe d'arbre fini étiqueté par $A \cup V \cup \{\varepsilon\}$ tel que pour tout nœud X non feuille, de fils $a_1, \dots, a_n$, alors $X \to a_1 \dots a_n$ est règle de G
La frontière d'un arbre de dérivation est la concaténation des étiquettes des feuilles de gauche à droite.
Si la frontière contient au moins une variable, l'arbre est un **arbre de dérivation partielle**. Sinon l'arbre est un **arbre de dérivation terminale/totale**.
Un mot w est engendré par une grammaire algébrique (G, S) ssi il est frontière d'un arbre de dérivation de (G, S)
Un arbre de dérivation montre des dérivations possibles de $S \to^* w$ mais ne donne pas d'information sur l'ordre de ces dérivations. D'autres dérivations $S \to^* w$ peuvent exister avec un arbre distinct.
Il y a bijection entre les arbres de dérivation et les dérivations par la gauche (ou par la droite).

2.3.1. Ambiguïté

Une **grammaire algébrique $\boldsymbol{(G, S)}$ est ambigüe** ssi un de ses mots engendrés admet 2 arbres de dérivations distincts depuis S.
Dans une grammaire algébrique non ambigüe, un mot engendré càd une frontière correspond bijectivement à son arbre de dérivation.
Un langage algébrique est **non ambigu** ssi il est engendré par au moins une grammaire non ambigüe.

Un langage algébrique est **inhéremment ambigu** ssi toute grammaire qui l'engendre est ambigüe.
La grammaire $S \to SS + a$ est ambigue car aaa admet 2 arbres de dérivations. Le langage engendré par cette grammaire est a^+ n'est cependant pas ambigu car il est aussi engendré par la grammaire $S \to aS + a$ qui elle n'est pas ambigüe.

2.3.2. Lemme d'itération (Ogden)

Définitions: On distingue certaines feuilles d'un arbre (par exemple en leur ajoutant une étiquette valant 1 ou 0 suivant que la feuille est choisie distinguée ou non), un tel **arbre est à feuilles distinguables**.
Dans un arbre à feuilles distinguables, on dit qu'un **nœud est distingué** ssi le sous-arbre dont il est racine contient au moins une feuille distinguée. Dans un arbre à feuilles distinguables, on dit qu'un **nœud est spécial** ssi il a au moins deux fils distingués. Par définition un nœud spécial est distingué, le parent d'un nœud distingué est distingué. Un **arbre est de degré n** ssi chaque nœud a au plus n fils.
Un mot engendré d'une grammaire à lettres distinguables correspond à son arbre de dérivation à feuilles distinguables.
Lemme combinatoire pour preuve Ogden. Si chaque branche d'un arbre de degré m avec k feuilles distinguées, contient au plus r nœuds spéciaux, alors $k \leq m^r$. Par récurrence sur $r \geq 0$.
Lemme d'itération d'Ogden. Pour toute grammaire algébrique (G, S) il existe $K \in N$ tel que tout mot partiel engendré $f \in \widehat{L_G(S)}$ ayant au moins K lettres distinguées se factorise en $f = \alpha u \beta v \gamma$ où $\alpha, u, \beta, v, \gamma \in (A + V)^*$ avec :
1. $S \to^* \alpha T \gamma$ et $T \to^* uTv + \beta$
2. soit α, u et β, soit β, v et γ contiennent des lettres distinguées.
3. $u\beta v$ contient moins de K lettres distinguées.

Ces conditions montrent que tout mot de la forme $\alpha u^n \beta v^n \gamma, n \in N$ est un mot partiel engendré par (G, S). Une telle paire (u, v) est appelée **paire itérante**.
Le lemme d'Ogden est compliqué mais presque le seul outil qui permet de montrer que certains langages ne sont pas algébriques ou qu'ils sont inhéremment ambigus. Analogue du lemme de l'étoile pour les langages algébriques.
Dans le cas où toutes les lettres sont distinguées, on obtient le corollaire suivant.
Corollaire Bar-Hiller, Perles, Shamir. Pour tout langage algébrique L, il existe $N \in N$ tel que tout mot du langage de longueur $\geq N$ se factorise en $f = \alpha u \beta v \gamma$ avec $\alpha, u, \beta, v, \gamma \in (A + V)^*$, $|uv| > 0, |u\beta v| < N$ et $\forall n \in N \ \ \alpha u^n \beta v^n \gamma \in L$.

2.3.3. Applications du lemme d'itération

Le langage $\{a^n b^n c^n : n \in N\}$ n'est pas algébrique.
Corollaire : La classe des langages algébriques n'est ni close par intersection, ni par complémentation.
Le langage $\{a^m b^n c^m d^n : n, m \in N\}$ n'est pas algébrique.
Le langage $\{a^m b^m c^n : m, n \in N\} \cup \{a^m b^n c^n : m, n \in N\}$ est inhéremment ambigu.
Le langage $\{ww : w \in A^*\}$ n'est pas algébrique.
Le langage $\{w\#w : w \in A^*\}$ avec $\# \notin A$, n'est pas algébrique.
Le langage $\{w\#w' : w, w' \in A^*, w \neq w', |w| = |w'|\}$ avec $\# \notin A$ n'est pas algébrique.

2.3.4. Ambiguïté inhérente

Difficile de montrer l'ambiguïté inhérente d'un langage algébrique. Lorsque le lemme d'itération ne marche pas on utilise la fonction génératrice.
La **fonction génératrice d'un langage L** est $f_L(z) = \sum_{n=0}^{\infty} a_n z^n$ avec $a_n = |L \cap A^n|$
La fonction génératrice d'un langage algébrique non ambigu est algébrique car solution du système commutatif d'une grammaire algébrique non ambigu qui engendre le langage.
Flageolet. Le langage de Goldstine est inhéremment ambigu.

2.4. Propriétés de clôture

2.4.1. Operations rationnelles

Les langages algébriques sont clos par les opérations rationnelles union, concaténation, étoile. Mais pas

par intersection ou complémentation.
Les langages rationnels sont aussi algébriques. (corollaire ou preuve directe via automates normalisés).

2.4.2. Substitution algébrique

Rappel : Une **substitution de A^* vers B^*** correspond à un morphisme de monoïdes $\sigma: (A^*,\cdot) \to (P(B^*),\cdot)$
Une **substitution σ de A^* vers B^* est algébrique** ssi $\sigma(a)$ est un langage algébrique pour tout $a \in A$.
Un morphisme de mots $\sigma: A^* \to B^*$ correspond en particulier à une substitution algébrique.
L'image d'un langage algébrique par un morphisme de mots est encore un langage algébrique.
L'image d'un langage algébrique L par une substitution algébrique σ est $\sigma(L) = \bigcup_{w \in L} \sigma(w)$ encore un langage algébrique.

2.4.3. Intersection avec un rationnel

L'intersection d'un langage algébrique avec un langage rationnel est un langage algébrique (donc aussi rationnel). (preuve par automates ou par monoïdes).

2.4.4. Morphisme inverse

Un **morphisme de mots $A^* \to B^*$ est alphabétique (resp. strictement)** ssi $\forall a \in A\ |\sigma(a)| \leq 1$ (resp.$= 1$)
Lemme factorisation d'un morphisme. Pour un morphisme de mot $h: A^* \to B^*$ il existe 2 morphismes alphabétiques $g: C^* \to B^*, \pi: C^* \to A^*$ et un langage rationnel $K \subseteq C^*$ tel que $\forall w \in B^*\ h^{-1}(w) = \pi(g^{-1}(w) \cap K)$
Théorème morphisme inverse. L'image réciproque d'un langage algébrique par un morphisme de mot est encore un langage algébrique.

2.4.5. Théorème de Chomsky et Schutzenberger

Chomsky et Schutzenberger. Un langage L est algébrique ssi $L = \varphi(D_n^* \cap K)$ avec $n \in N,\ K$ un langage rationnel, et φ un morphisme alphabétique.

2.5. Forme normale de Greibach

Une **grammaire algébrique $G = (A, V, P)$ est en forme normale de Greibach** ssi chacune de ses règles est de la forme $S \to w$ avec $w \in AV^*$. Si de plus chaque $w \in A + AV + AV^2$, la **grammaire algébrique est en forme normale de Greibach quadratique**.
Greibach. Toute grammaire algébrique propre est équivalente à une grammaire en forme normale de Greibach.
Rosenkrantz. Toute grammaire algébrique propre est équivalente à une grammaire en forme normale de Greibach quadratique.
Il existe une version encore plus affinée par Holtz si les membres droits $w \in A + AA + AVA + AVVA$.

2.6. Automates à pile

Les automates à pile sont une extension des automates finis. Ils ont en plus une mémoire auxiliaire organisée sous la forme d'une pile. On ne peut qu'empiler ou dépiler des symboles, seul le symbole en haut de la pile est visible. Les transitions ne dépendent que de l'état, du mot lu, et du symbole en haut de la pile. Les langages acceptés sont exactement les langages algébriques.
L'alphabet d'entrée est souvent supposé contenir le mot vide (automate à ε- transitions). Pour un automate à pile non déterministe il est possible de les supprimer mais ce n'est pas immédiat, par contre pour les automates à pile déterministe, ces ε-transitions sont indispensables.
Dans une transition, on dépile le symbole en haut de la pile, puis on peut empiler une chaine (pas seulement un symbole) sur l'alphabet de pile en une seule transition, ε indique qu'on empile rien.

2.6.1. Définitions et exemples

Un **automate à pile (fini, non déterministe)** correspond à $(Q, A, Z, E, q_0, z_0, \mathrm{m})$ avec A **alphabet d'entrée** fini, un **alphabet de pile** fini Z, Q ensemble fini d'**états**, $\boldsymbol{q_0} \in Q$ **l'état initial**, $\boldsymbol{z_0} \in Z$ **symbole de pile initial**, $E \subseteq (Q \times Z) \times (A \cup \{\varepsilon\}) \times (Q \times Z^*)$ ensemble de **transitions** notées $\boldsymbol{q, z \to^a q', h}$, m un **mode d'acceptation** (plusieurs sont possibles, on les définit plus loin).
Une autre façon de modéliser les transitions est de remplacer E par une **fonction de transition** $\boldsymbol{\delta}: (Q \times Z) \times (A \cup \{\varepsilon\}) \to P(Q \times Z^*): (q, z, a) \mapsto \{(q', hx) \mid (q, z) \to^a (q', h) \in E\}$.

Une **configuration externe d'un automate à pile $\boldsymbol{M}$** correspond à un (q, p, u, v) avec $q \in Q$ un **état courant**, $p \in Z^*$ le **contenu courant de la pile**, $u \in A^*$ les lettres lues du mot scanné, $v \in A^*$ les lettres restantes à lire du mot scanné.
Une **configuration interne d'un automate à pile $\boldsymbol{M}$** correspond juste à un (q, p) avec $q \in Q$ et $p \in Z^*$
Calcul. On définit la relation $\vdash_{\mathbf{M}}$ et on note une **étape de calcul** comme correspondant à un couple de configurations et une lettre $a \in A$ correspondant à une transition. On peut la qualifier d'externe et la noter $\boldsymbol{C} = (\boldsymbol{q}, \boldsymbol{zx}, \boldsymbol{u}, \boldsymbol{av}) \vdash_{\mathbf{M}} \boldsymbol{C'} = (\boldsymbol{q'}, \boldsymbol{hx}, \boldsymbol{ua}, \boldsymbol{v})$ si on considère dans le contexte un mot scanné fixé, ou d'interne sinon, auquel cas on peut l'identifier à la transition $(q, z) \rightarrow^a (q', h) \in E$.
Un chemin/une exécution dans un automate à pile $\boldsymbol{M}$ correspond à une suite consécutive finie d'étapes de calcul. On peut la qualifier d'externe et la noter $\boldsymbol{C_0} = (\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{u}, \boldsymbol{wv}) \vdash^*_{\mathbf{M}} \boldsymbol{C_n} = (\boldsymbol{q'}, \boldsymbol{p'}, \boldsymbol{uw}, \boldsymbol{v})$ si on considère un mot particulier, ou d'interne sinon, auquel cas on peut noter $(\boldsymbol{q}, \boldsymbol{p}) \rightarrow^{\boldsymbol{w}} (\boldsymbol{q'}, \boldsymbol{p'})$.
Une **configuration initiale d'un automate à pile $\boldsymbol{M}$** est de la forme $(q_0, z_0, w \in A^*, \varepsilon)$
Modes d'acceptation. Une **configuration finale d'un automate à pile $\boldsymbol{M}$** correspond à une configuration particulière de l'automate d'un des modes suivants choisis comme l'unique **mode d'acceptation** :
Une **configuration finale de type état final** : $q \in F$ avec $F \subseteq Q$ **ensemble d'états finaux**.
Une **configuration finale de type pile vide** : $p = \varepsilon$
Une **configuration finale de type fond de pile testable** : $p = z_F p'$ et $z_F \in Z_F \subseteq Z$ symbole de fin de pile.
Une **combinaison de ces 3 modes**. (Fin de la définition d'automate a pile).
Une exécution d'un automate à pile $\boldsymbol{M}$ est acceptante ssi sa configuration de départ C_0 est initiale et sa configuration d'arrivée C_n est finale.
Un **mot $\boldsymbol{w}$ est accepté/reconnu** par un automate à pile M s'il est l'étiquette d'une exécution acceptante.
Le langage (accepté) d'un automate à pile $\boldsymbol{M}$, est l'ensemble $\boldsymbol{L(M)}$ des mots de A^* qu'il accepte.
Deux automates à pile sont **équivalents** ssi ils acceptent le même langage.
Un automate à pile fini peut être vu comme un automate infini définissant un graphe (multiple) infini, en prenant pour états les configurations internes. Mais les automates infinis ont peu d'intérêt car trop puissants. Ce n'est pas possible de représenter un automate à pile avec un automate car la mémoire utilisée par la pile est illimitée càd que le nombre de configurations internes possibles est infini.
Les différents modes d'acceptation classiques sont équivalents (acceptent les mêmes langages).
Exemples d'automate à pile : Le langage des palindromes non vide est reconnu par un automate à pile d'acceptation par pile vide assez simple à construire.
Le langage de Dyck D_n^* peut être reconnu par un automate à pile.
L'automate à pile $E = \{q_0, z \rightarrow^a q_0, zzz\ ; q_0, z \rightarrow^\varepsilon q_1, \varepsilon\ ; q_1, z \rightarrow^b q_1 \varepsilon\}$ accepte le langage $\{a^n b^{2n} : n \in N\}$ pour le mode d'acceptation par pile vide, mais accepte le langage $\{a^n b^p : 0 \leq p \leq 2n\}$ pour le mode d'acceptation par état final $F = \{q_1\}$. L'équivalence des modes d'acceptations porte sur tous les automates à pile, mais pas sur un automate à pile particulier.
Intérêt des automates à piles d'acceptation a fond de pile: Un des problèmes est que le calcul se bloque dès que la pile devient vide ce qui peut rendre difficile la modélisation de certains algorithmes. Parfois si la pile est vide on préfère remettre quelque chose dedans pour continuer l'exécution. Cette difficulté peut être contournée en utilisant des automates à fond de pile testable qui peuvent tester si le sommet de pile est un symbole particulier z_F indiquant la fin de la pile.
Un **automate à compteur** est un automate à pile d'acceptation par fond de pile testable, tel qu'il n y a que 2 symboles de pile : le fond z_0 et un compteur z_1 donc $Z = \{z_0, z_1\}$ donc la pile est de la forme $z_1^* z_0$.
Une telle pile est utilisée comme un compteur représentant un entier positif ou nul. Le fond de pile permet de tester si la valeur du compteur est 0 sans pour autant arrêter l'exécution immédiatement.
Un **automate à pile est normalisé** ssi pour toute transition $q, z \rightarrow^a q', h$, le mot h qui remplace z sur la pile appartient à $\varepsilon + z + Zz$. Cela signifie que chaque transition peut soit dépiler un symbole, soit empiler un symbole, soit ne rien faire à la pile.
Tout automate à pile est équivalent à un automate à pile normalisé.

Tout automate à pile sans ε-transitions est équivalent à un automate à pile normalisé sans ε-transitions.

2.6.3. Equivalence avec les grammaires

Théorème.* Un langage est algébrique ssi c'est le langage d'un automate à pile.

Exemple sens direct : La construction de l'automate à pile dans ce théorème, appliquée à la grammaire des palindromes : $S \to aSa + bSb + a + b + \varepsilon$ donne:

$\{q, S \to^{\varepsilon} q, ASA\ ; q, S \to^{\varepsilon} q, A\ ; q, S \to^{\varepsilon} q, \varepsilon\ ; q, A \to^{a} q, \varepsilon\ ; q, S \to^{\varepsilon} q, BSB\ ; q, S \to^{\varepsilon} q, B\ ; q, B \to^{b} q, \varepsilon\}$

avec S symbole de pile initial, q état initial (unique état). Mode d'acceptation par pile vide.

Exemple pour la réciproque : La méthode des triplets de Ginsburg appliquée à l'automate à pile acceptation par pile vide : $\{q_0, z \to^{a} q_0, zzz\ ; q_0, z \to^{\varepsilon} q_1, \varepsilon\ ; q_1, z \to^{b} q_1\varepsilon\}$, fournit la grammaire $\{R \to aRRR, S \to aRRS + aRST + aSTT + \varepsilon, T \to b\}$ ou R, S, T correspondent aux triplets $(q_0, q_0, z), (q_0, q_1, z), (q_1, q_1, z)$. Apres réduction et substitution de T par b, on obtient la grammaire $\{S \to aSbb + \varepsilon\}$ qui engendre bien le langage $\{a^n b^{2n} : n \in N\}$.

Une **grammaire algébrique est linéaire** ssi toute règle $X \to w$ vérifie $w \in A^*VA^* + A^*$ autrement dit ssi tout membre droit d'un règle comporte au plus une variable.

Une **grammaire est linéaire droite** ssi toutes ses productions font apparaître la variable tout à droite.

Un **langage est linéaire** ssi il est engendré par une grammaire algébrique linéaire.

L'image par un morphisme de mots d'un langage linéaire est encore un langage linéaire.

L'intersection d'un langage linéaire avec un langage rationnel est un langage linéaire.

Un **automate à pile est à pic** ssi dans toute exécution, la taille de la pile ne fait d'abord que croitre jusqu'à une certaine configuration, puis ne fait que décroitre jusqu'à la dernière configuration.

Un langage est linéaire ssi c'est le langage d'un automate à pic.

Caractérisation langage rationnel.

Un langage est rationnel est engendré par une grammaire linéaire droite non ambigüe.

Une union disjointe de langages non ambigus est un langage non ambigu.

2.6.4. Automates à pile déterministes

Un **automate à pile est déterministe** ssi sa fonction de transition δ est partielle $\delta: (Q \times Z) \times (A \cup \{\varepsilon\}) \to Q \times Z^*$ et ((q, z, ε) définie $\Rightarrow \forall a \in A\ \delta(q, z, a)$ pas définie).

Autrement dit ssi : pour tout $(q, z) \in Q \times Z$:

- soit il existe une unique transition de la forme $q, z \to^{\varepsilon} q', h$ et il n'existe aucune transition de la forme $q, z \to^{a} q', h$
- soit il n'existe pas de transition de la forme $q, z \to^{\varepsilon} q', h$ et pour chaque lettre $a \in A$ il existe au plus une transition de la forme $q, z \to^{a} q', h$

L'automate à pile $\{q_0, z \to^{a} q_0, zzz\ ; q_0, z \to^{\varepsilon} q_1, \varepsilon\ ; q_1, z \to^{b} q_1\varepsilon\}$ accepte par pile vide le langage $\{a^n b^{2n} : n \in N\}$ mais n'est pas un automate déterministe.

Le langage $\{a^n b^{2n} : n \in N\}$ est accepté par pile vide par l'automate à pile déterministe :

$\{q_0, z \to^{a} q_1, zz\ ; q_1, z \to^{a} q_1, zzz\ ; q_1, z \to^{b} q_2, \varepsilon\ ; q_2, z \to^{b} q_2, \varepsilon\}$

Dans le cas des automates à pile <u>déterministes</u> les différents modes d'acceptations ne sont plus équivalents. L'acceptation par pile vide permet seulement d'accepter des langages préfixes, c'est-à-dire des langages tels que 2 mots du langage ne sont jamais préfixes l'un de l'autre.

Un **langage algébrique est déterministe** ssi c'est le langage d'un automate à pile déterministe.

Le complémentaire d'un langage algébrique déterministe est un langage algébrique déterministe.

Le complémentaire d'un langage algébrique n'est pas nécessairement algébrique.

Les langages algébriques déterministes forment donc une sous-classe stricte des langages algébriques.

Tout langage algébrique déterministe est non ambigu.

Réciproque fausse : le langage algébrique $\{a^n b^n : n \in N\} \cup \{a^n b^{2n} : n \in N\}$ est non ambigu et non déterministe.

2.7. Compléments

2.7.1. Réécriture

Dans un contexte ou $\rightarrow$ symbolise une réduction (par exemple les grammaires formelles).
Un **élément $x \in X$ est irreductible/en forme normale pour une relation binaire $\rightarrow$ sur X** ssi il n'existe pas de $y \in X$ tel que $x \rightarrow y$
On note $\rightarrow^*$ la cloture reflexive transitive d'une relation binaire $\rightarrow$. On dit **x se réduit en y** pour $x \rightarrow^* y$
Une **relation binaire $\rightarrow$ est noethérienne** ssi il n'existe pas de chaine infinie $x_0 \rightarrow x_1 \rightarrow \cdots$
Pour une relation $\rightarrow$ noethérienne, tout élément se réduit en au moins un élément en forme normale.
Une **relation binaire $\rightarrow$ est confluente/vérifie la propriété diamant** ssi $\begin{cases} x \rightarrow^* y \\ x \rightarrow^* y' \end{cases} \Rightarrow \exists z \in X \begin{cases} y \rightarrow^* z \\ y' \rightarrow^* z \end{cases}$
Pour une relation binaire $\rightarrow$ confluente, tout élément se réduit en au plus un élément en forme normale.
La confluence est souvent difficile à vérifier, on introduit 2 propriétés plus faciles.
Une **relation binaire $\rightarrow$ est fortement confluente** ssi $\begin{cases} x \rightarrow^* y \\ x \rightarrow^* y' \end{cases} \Rightarrow \exists z \in X \begin{cases} y = z \text{ ou } y \rightarrow z \\ y' \rightarrow^* z \end{cases}$
Une **relation binaire $\rightarrow$ est localement confluente** ssi $\begin{cases} x \rightarrow y \\ x \rightarrow y' \end{cases} \Rightarrow \exists z \in X \begin{cases} y \rightarrow^* z \\ y' \rightarrow^* z \end{cases}$
La forte confluence entraine la confluence.
La confluence locale n'entraine pas la confluence en général.
Newman. Une relation noethérienne et localement confluente est confluente.
Pour une relation binaire confluente et noethérienne, tout élément se réduit en exactement un élément en forme normale.

2.7.2. Contenus de piles

Pour un automate à pile (d'acceptation par fond de pile) $(Q, A, Z, z_0, E, q_0, F)$, le langage de tous les contenus de piles possibles $H = \{h \in Z^* \mid \exists f \in A^* \, \exists q \in Q \; (q_0, z_0) \rightarrow^f (q, h)\}$ est rationnel.*

2.7.3. Groupe libre

Benoit 1969. Une partie d'un groupe libre d'alphabet $A = \{a_1, \dots, a_n, a_1^{-1}, \dots, a_n^{-1}\}$ est rationnelle ssi son image directe dans A^* par l'injection canonique du groupe libre vers A^* est rationnelle dans A^*.
Corollaire. La famille des parties rationnelles d'un groupe libre est close pour les opérations booléennes.
Un sous-groupe est une partie rationnelle de son groupe ssi il de type fini.
Howson. L'intersection finie de sous-groupes de types finis d'un groupe libre, est un sous-groupe de type fini.