909

Langages rationnels. Exemples et applications

[CAR] 1.5

Motivations :
- Classe très robuste, donc très manipulable
- Lien avec les automates, modèle simple d'ordinateur

I) Définitions, premières propriétés

1) Langages rationnels

Def 1 : Σ un alphabet. Un langage sur Σ est un ensemble de mots formés de lettres de Σ.

Def 2 : On définit les opérations rationnelles sur L et L' deux langages :

$L + L' = L \cup L'$ $\quad LL' = \{uv \mid u \in L, v \in L'\}$

$L^i = \underbrace{LL \dots L}_{i \text{ fois}}$ $\quad L^* = \bigcup_{i \geq 0} L^i$ $\quad L^+ = LL^*$

$L^0 = \{\varepsilon\}$

Def 3 : La classe $Rat(\Sigma)$ des langages rationnels sur Σ est la plus petite famille de langages telle que :
1) $\emptyset \in Rat(\Sigma)$; $\forall a \in \Sigma$, $\{a\} \in Rat(\Sigma)$
2) $Rat(\Sigma)$ est close par union, produit, étoile

Ex 4 : $\{\varepsilon\}$; Σ ; Σ^* ; $\{u \mid |u| \text{ est paire}\}$ sont rationnels

On peut également représenter les langages rationnels comme une classe d'expressions, dites expressions rationnelles :

Def 5 La classe $ER(\Sigma)$ des expressions rationnelles est la plus petite famille d'expressions telles que
* $\emptyset \in ER(\Sigma)$, $\forall a \in \Sigma$, $a \in ER(\Sigma)$
* $\forall E, E' \in ER(\Sigma)$, $\{E+E', E.E', E^*\} \subset ER(\Sigma)$

Ex 6 : $a\Sigma\Sigma$ représente les mots de longueur 3 qui commencent par a
$(ab^*a + b)^*$: mots ayant un nombre pair de a. ($\Sigma = \{a,b\}$)

2) Automates finis

Rappel 7 : Un automate fini est un quintuplet $(Q, \Sigma, I, F, \delta)$ avec Q fini (états), $I \subseteq Q$ états initiaux, $F \subseteq Q$ états finaux, δ fonction de transition

Not : Pour $\mathcal{A}$ automate, $\mathcal{L}(\mathcal{A})$ est le langage reconnu par $\mathcal{A}$

Ex 8 : 

$= \mathcal{A}$. $\mathcal{L}(\mathcal{A}) = (ab^*a + b)^*$

Thm 8 [Kleene] :
Un langage L est rationnel si et seulement si il existe un automate fini $\mathcal{A}$ tel que $L = \mathcal{L}(\mathcal{A})$
Preuve : On exhibe un algorithme pour chaque implication

DEV 1 [CAR] [SAK]

Rem 9 : Thm 8 implique que $Rat(\Sigma)$ est stable par complémentaire et intersection.

Une conséquence importante du théorème de Kleene est :

Lemme 10 : [Lemme de l'étoile simple] [CAR] 1.9
Si L est un langage rationnel, alors il existe $N \in \mathbb{N}$ tel que :

$$\forall m \in L,\ |m| \geq N \Rightarrow \begin{cases} \exists u,v,w \in \Sigma^*,\ v \neq \varepsilon,\ |v| \leq N \text{ t.q.} \\ m = uvw \text{ et } \forall k \in \mathbb{N},\ uv^kw \in L \end{cases}$$

Utilité : Ce lemme sert principalement à prouver la non-rationnalité de certains langages.
Par exemple, $\{a^{2^k} \mid k \in \mathbb{N}\}$ n'est pas rationnel

[CAR] 1.9

II) Autres caractérisations

1) Lemme de l'étoile fort

Lemme 11 : [Lemme de l'étoile fort]
Si L est un langage rationnel, alors il existe $N \in \mathbb{N}$ tel que
$$\left.\begin{matrix} m = u\, v_1 v_2 \dots v_N\, w \\ m \in L\ ;\ v_1, \dots, v_N \in \Sigma^+ \end{matrix}\right\} \Rightarrow \left\{\begin{matrix} \exists i,j\ \ 0 \leq i < j \leq n, \\ \forall k \in \mathbb{N},\ u\, v_1 \dots v_{i-1} (v_i \dots v_j)^k v_{j+1} \dots v_N w \in L \end{matrix}\right.$$

Rem 12 : Ce lemme est plus fort que le précédent. Par exemple, il permet de prouver que $\{a^n b^n \mid n \in \mathbb{N}\} \cup A^* b a A^*$ n'est pas rationnel, alors qu'il vérifie le lemme simple

Rem 13 : Le lemme 11 n'est pas une condition suffisante de rationalité.

C-Ex 14 : $\{a^{k_1} b a^{k_2} b \dots a^{k_n} b \mid n \in \mathbb{N}, \exists i \leq n,\ i \neq k_i\}$ vérifie le lemme 11, mais n'est pas rationnel

Caractérisation 15 : [ADMIS]
$L \in Rat(\Sigma) \iff L$ et $\bar{L}$ vérifie le lemme de l'étoile fort

[CAR] 1.7.1

2) Théorème de Myhill - Nérode

Def 16 : Le résiduel de L par $u \in \Sigma^*$ est :
$$u^{-1}L = \{v \in \Sigma^* \mid uv \in L\}$$

Thm 17 [Myhill-Nérode] : L un langage
$$L \in Rat(\Sigma) \iff \{u^{-1}L \mid u \in \Sigma^*\} \text{ est fini}$$

Preuve : $\Leftarrow$ Thm de Kleene
$\Rightarrow$ Construction de l'automate des résiduels (qui reconnait L)
$(Q_L, \Sigma, \{i_L\}, F_L, \delta_L)$ avec
$Q_L = \{u^{-1}L \mid u \in \Sigma^*\}$ $\quad i_L = \varepsilon^{-1}L = L$
$F_L = \{u^{-1}L \mid \varepsilon \in u^{-1}L\}$ $\quad \delta_L(u^{-1}L, a) = (ua)^{-1}L$

3) Reconnaissance par grammaire régulière

Def 18 : Une grammaire régulière est un quadruplet (V, Σ, R, S)
+ V est un alphabet | Σ : ensemble des symboles terminaux
+ $\Sigma \subsetneq V$ | $V \setminus \Sigma$: ensemble des symboles non-terminaux
+ $R \subset (V \setminus \Sigma \times \Sigma^*.(V \setminus \Sigma)) \cup (V \setminus \Sigma \times \Sigma^*)$
règles de la forme $A \to aB$ ou $A \to a$
+ $S \in V \setminus \Sigma$ symbole de départ

Le langage généré est l'ensemble des mots sur Σ obtenus par applications successives de règles de R à partir de S.

Prop 19 : Un langage est rationnel si et seulement si il est le langage généré par une grammaire régulière

DEV 2

[WOL] p.51

4) Reconnaissance par monoïde

[CAR] 1.11.

Prop 20 : Si Σ est un alphabet, Σ^* est un monoïde pour la concaténation

Rem 21 : Un morphisme de monoïde sur Σ^* est entièrement déterminé par l'image des lettres

Def 22 : $L \subseteq \Sigma^*$ est dit reconnu par un morphisme $\mu : \Sigma^* \to M$ si il existe $P \subset M$ telle que $L = \mu^{-1}(P)$.
Si un tel morphisme existe, on dit que M reconnait L

Prop 23 : $L \in Rat(\Sigma) \iff L$ reconnu par un monoïde fini

III) Outils théoriques

1) Cloture par morphisme

[CAR] 1.8.2

Prop 24 : Soit $\mu : \Sigma^* \to A^*$ morphisme de monoïde. Alors :
* $L \in Rat(\Sigma) \Rightarrow \mu(L) \in Rat(A)$
* $K \in Rat(A) \Rightarrow \mu^{-1}(K) \in Rat(\Sigma)$

Rem 25 : C'est encore une propriété de stabilité de $Rat(\Sigma)$, preuve de la grande robustesse de cette classe.

[CAR] 1.5 p.40

2) Lemme d'Arden

Lemme 26 [Lemme d'Arden]: Soient K et L deux langages, et l'équation $X = KX + L$.
Alors: * $\varepsilon \notin K \Rightarrow$ il y a une unique solution $X = K^*L$
* $\varepsilon \in K \Rightarrow$ les solutions sont de la forme $X = K^*(L+Y)$ avec $Y \subseteq A^*$

Application: Méthode de calcul pour trouver le langage reconnu par un automate $\mathcal{A} = (Q, \Sigma, I, F, \delta)$.
$\forall p \in Q$; X_p est le langage des mots acceptés avec p comme état initial
On résout le système $\left(X_p = \begin{cases} \sum_{(p,a,q) \in \delta} a X_q + \varepsilon & \text{si } p \in F \\ \sum_{(p,a,q) \in \delta} a X_q & \text{sinon} \end{cases}\right)_{p \in Q}$
par élimination gaussienne et alors
$$\mathcal{L}(\mathcal{A}) = \sum_{i \in I} X_i$$

DEV3 [PET]

3) Sous-mots

Def 27: Une antichaine sur Σ est un langage formé de mots dont aucun n'est sous-mot d'un autre.

Thm 28 [Higman]: Toute antichaine est finie

Cor 29: $\forall L \subset \Sigma^*$, Sous-Mots$(L) \in Rat(\Sigma)$.

4) Un peu de complexité

* Savoir si le langage reconnu par un automate est vide en est $O(k+m)$, $k = |Q|$, m le nombre de transitions
→ Parcours de graphe pour déterminer si il y a un état final accessible depuis un état initial

* Savoir si le langage reconnu par un automate est infini est aussi en $O(k+m)$
→ Recherche d'un cycle accessible et co-accessible

V) Applications

1) Recherche de motifs

On se donne $P \in \Sigma^*$ motif, $T \in \Sigma^*$ texte.

Objectif: Déceler les occurrences de P dans T, $|P| = m$; $|T| = n$.

Def 29: Pour $k \leq m$, on pose P_k le préfixe de taille k de P

Def 30: Pour $x \in \Sigma^*$, on pose $\sigma(x) = \max\{k \mid P_k \text{ suffixe de } x\}$

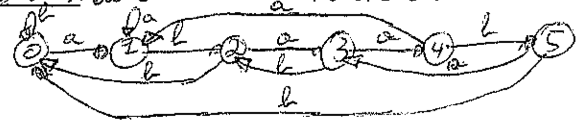
[COR] 916

Def 31: L'automate de recherche associé à P est défini par: $\mathcal{A} = (Q, \Sigma, I, F, \delta)$
$Q = \{0, 1, \dots, m\}$
$I = \{0\}$
$F = \{m\}$
$\forall q \in Q, a \in \Sigma$, $\delta(q, a) = \sigma(P_q a)$.

Prop 32: $\mathcal{L}(\mathcal{A}) = \Sigma^* P$.

Rem 33: $\mathcal{A}$ est minimal

Ex 34: Automate de recherche de a b a a b



[CAR] 3.8

2) Arithmétique de Presburger

Def 35: L'arithmétique Presburger est une théorie logique du premier ordre des entiers munis de l'addition.

Ex 36: $\forall x \; \exists y \; x = y + y$ et $\forall x \; x = x$ sont des formules de l'arithmétique Presburger

Def 37: Une théorie logique est dite décidable s'il est décidable de savoir si une formule close est vraie

Thm 38: L'arithmétique de Presburger est décidable.

Preuve: Utilisation d'automates
- Cas de base: automates de l'addition et de l'égalité
- Hérédité par opérations rationnelles sur les automates et projections

Wolper
p. 51-52

Grammaires régulières

ε !!!!!!!!

Commençons par un rappel sur les grammaire :

Définition : Une grammaire est un quadruplet $G = (V, \Sigma, R, S)$ avec :

- V est un alphabet
- $\Sigma \subset V$ est l'ensemble des symboles terminaux.
  $V - \Sigma$ est alors l'ensemble des symboles non-terminaux
- $R \subset (V^+ \times V^*)$ est un ensemble fini de règles (notées $\underset{\in V^+}{\alpha} \underset{G}{\rightarrow} \underset{\in V^*}{\beta}$)
- $S \in V \setminus \Sigma$ est le symbole de départ

Dérivation : $u \in V^+$, $v \in V^*$. v dérive de u en une étape ssi :

$u = x u' y$; $v = x v' y$; $u' \underset{G}{\rightarrow} v'$ (c-à-d que la règle (u', v') est dans R)

On note $u \underset{G}{\Rightarrow} v$

Si v dérive de u en plusieurs étapes, on note $u \underset{G}{\overset{*}{\Rightarrow}} v$

Langage généré : Un mot $v \in \Sigma^*$ est dit généré par G si $S \underset{G}{\overset{*}{\Rightarrow}} v$.

$L(G)$ le langage généré par G est $\{v \in \Sigma^* \mid S \underset{G}{\overset{*}{\Rightarrow}} v\}$

Grammaires régulières : Une grammaire est dite régulière si toutes ses règles sont de la forme $A \rightarrow wB$ ou $A \rightarrow w$ avec $A, B \in V \setminus \Sigma$, $w \in \Sigma$

Si l'on considère une dérivation sur une grammaire régulière, à tout moment de la dérivation, il n'y a qu'un seul symbole non-terminal ou aucun dans la dérivation.

L'objectif est de montrer :

> **Théorème :** Un langage est rationnel si et seulement si il est généré par une grammaire régulière

Preuve :

Rationel $\Rightarrow$ généré par grammaire régulière :

Soit L rationel. Par théorème de Kleene, L est reconnu par un automate fini $M = (Q, \Sigma, s, F, \delta)$. $L = L(M)$.

Nous allons exhiber une grammaire grâce à M :

Soit $G = (V_G, \Sigma_G, R_G, S_G)$ avec :

- $\Sigma_G = \Sigma$ (les terminaux de G sont l'alphabet de M)
- $V_G = Q \cup \Sigma$ (les non-terminaux sont les états de M)
- $S_G = s$
- $R_G = \{A \to wB \mid \delta(A, w) = B\} \cup \{A \to \varepsilon \mid A \in F\}$

H_n : " $\forall p, q \in Q ; u, v \in \Sigma^*$; $(q, uv) \underset{M}{\to}^* (p, v) \iff q \underset{G}{\Rightarrow}^* uv$ si la dérivation est de longueur n. (Et alors l'exécution de M est aussi de longueur n)"

Montrons le résultat par récurrence sur n. (récurrence forte)

H_1 : Par définition de G, une dérivation d'une étape correspond à une exécution de M d'une étape.

Soit n fixé. Supposons H_k pour $k \in [\![1, n-1]\!]$

Soit $u, v \in \Sigma^*$, $p, q \in Q$

$$(q, uv) \underset{M}{\to}^* (p, v) \text{ en } n \text{ étapes} \iff \begin{cases} (q, uv) \underset{M}{\to}^* (q', u_n v) \text{ en } n-1 \text{ étapes} \\ \text{et } (q', u_n v) \underset{M}{\to} (p, v) \quad q' \in Q \end{cases}$$

$$\underset{H_1 \text{ et } H_{n-1}}{\iff} \begin{cases} q \underset{G}{\Rightarrow}^* u_{[1 \dots n-1]} q' \text{ en } n-1 \text{ étapes} \\ q' \underset{G}{\Rightarrow}^* u_n p \end{cases}$$

$$\iff (q, uv) \underset{G}{\Rightarrow}^* (p, v) \text{ en } n \text{ étapes}$$

d'où H_n

On conclut par récurrence que :

$$\forall p, q \in Q, u, v \in \Sigma^* \quad \underline{(q, uv) \underset{M}{\to}^* (p, v) \iff q \underset{G}{\Rightarrow}^* up}$$

Pour $q = s$; $p \in F$; $v = \varepsilon$, on obtient :

$(s, w) \underset{M}{\vdash}^* (p, \varepsilon) \iff s \underset{G}{\Rightarrow}^* w$, ce qui prouve la première implication

Généré par grammaire régulière $\Rightarrow$ Rationel

Soit L généré par $G = (V_G, \Sigma_G, R_G, S_G)$.

On pose $M = (Q, \Sigma, s, F, \delta)$ l'automate fini :

- $Q = V_G \setminus \Sigma_G \cup \{f\}$ avec f un nouvel état
- $\Sigma = \Sigma_G$
- $s = S_G$
- $F = \{f\}$
- $\delta : (A, w) \mapsto \{B \mid A \to wB \in R_G\} \cup \{f \text{ si } A \to w \in R_G\}$

On peut prouver de façon ~~quasi~~-identique à la première implication que

$$(q, uv) \xrightarrow[M]{*} (p, v) \Leftrightarrow q \overset{*}{\underset{G}{\Rightarrow}} up \quad \text{pour } p \neq f.$$

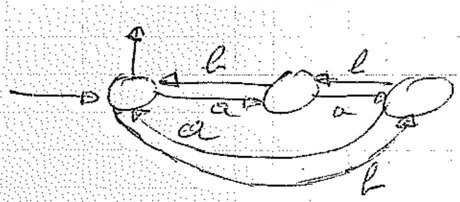
et donc que :

$$\forall w \in \Sigma^*, \quad (s, w) \xrightarrow[M]{*} (f, \varepsilon) \Leftrightarrow s \overset{*}{\underset{G}{\Rightarrow}} w.$$

D'où la réciproque $\square$

Exemple : $\quad L = \{w,\ |w|_a \equiv |w|_b\ [3]\}$ avec $\Sigma = \{a, b\}$

Automate :



Grammaire :

Symbole de départ : X_0

Règles :

$X_0 \to a X_1$

$X_0 \to b X_2$

$X_1 \to a X_2$

$X_1 \to b X_0$

$X_2 \to a X_0$

$X_2 \to b X_1$

$X_0 \to \varepsilon$

leene

Théorème de Kleene

Théorème.
- Soit E une expression rationnelle
  On peut construire un automate $\mathcal{A}$ tel que $\mathcal{L}(E) = \mathcal{L}(\mathcal{A})$.
- Soit $\mathcal{A}$ un automate
  On peut construire une expression rationelle E telle que $\mathcal{L}(E) = \mathcal{L}(\mathcal{A})$

Algo 1 : $E \to \mathcal{A}$: Méthode de Thompson.

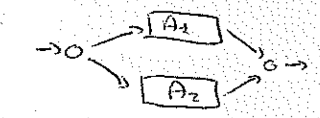
On construit inductivement un automate normalisé reconnaissant $\mathcal{L}(E)$.

$\emptyset : \to O \quad O \to$

$a \in \Sigma : \to O \xrightarrow{a} O \to$

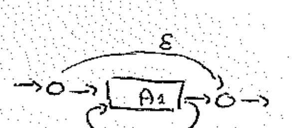
. Soit $\to O \to \boxed{A_1} \to O \to$ et $\to O \to \boxed{A_2} \to O \to$ reconnaissant E et F.

$E+F$:



$E.F : \to O \to \boxed{A_1} \to O \to \boxed{A_2} \to O \to$

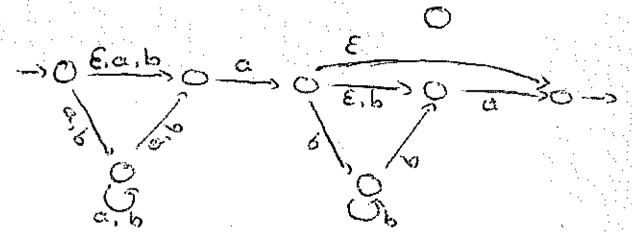
E^* :



: On copie les transitions vers l'état final en les faisant aboutir sur le nouvel état. On fait partir de celui-ci une copie des transitions issues de l'état initial.

On considère l'expression
$E = (a+b)^* a (ba^* + \varepsilon)$

L'algorithme fournit l'automate :



Correction 1.

Elle est assurée par les définitions.

Algo 2 : $\mathcal{A} \to E$: Elimination de Brzozowski

On considère des automates à expressions rationnelles normalisés : un tel automate A possède un unique état initial sans transition entrante, un unique état final sans transition sortante, et ses arêtes sont étiquetées par des expressions rationnelles.

Soit $\mathcal{A}$ un automate à n états.

Initialisation : Il existe un automate A_0 à expressions rationnelles et à $n+2$ états tel que $\mathcal{L}(A_0) = \mathcal{L}(\mathcal{A})$

Invariant de boucle : Soit A_k ayant $n+2-k$ états, il existe un automate A_{k+1} ayant $n+1-k$ états tel que $\mathcal{L}(A_{k+1}) = \mathcal{L}(A_k)$

Terminaison : On calcule $A_0, A_1 .. A_n$ et on s'arrête.

Correction : A_n est de la forme $\to (i) \xrightarrow{E} (f) \to$ Alors $\mathcal{L}(A_n) = E$ et $\mathcal{L}(A_n) = \mathcal{L}(\mathcal{A})$ □

Note : $\mathcal{L}(A) = \{ w_1 = u_1 \cdots u_k \in \Sigma^* \,/\, \exists\, i \xrightarrow{E_1} q_1 \cdots q_{k-1} \xrightarrow{E_k} f$ dans A et $u_i \in E_i$ pour $i \in [\![1, k]\!] \}$

1) Initialisation : On ajoute à $\mathcal{A}$ des états i et f, et on remplace les transitions de $\mathcal{A}$ par des transitions à étiquette rationnelle :

Pour $q,p \in Q^2$, $E^0_{qp} := \sum a \,/\, q \xrightarrow{a} p$ existe dans $\mathcal{A}$. Eventuellement, $E_{qp} = \emptyset$!!

Pour $p \in Q$, $E^0_{ip} := \sum a \,/\, \exists j \in I$ ~~tq~~ $i \xrightarrow{a} p$ existe dans $\mathcal{A}$

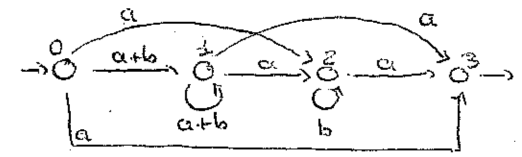
$E^0_{pf} := \sum a \,/\, \exists t \in F$ tq $p \xrightarrow{a} t$ existe dans $\mathcal{A}$.

Alors $w \in \mathcal{L}(A) \iff w = u_1 \dots u_k$, $u_i \in E_i$ avec $i \xrightarrow{E_1} \dots \xrightarrow{E_k} f$ dans A

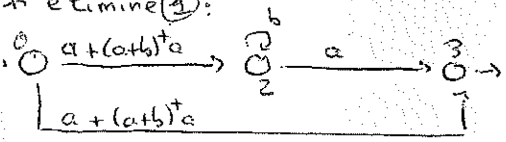
$\iff \exists j \in I, \exists t \in F$, $j \xrightarrow{u_1} \dots \xrightarrow{u_k} t$ dans $\mathcal{A}$ Ici $|u_i| = 1$

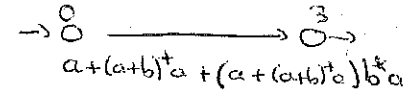
$\iff w \in \mathcal{L}(\mathcal{A})$. $\square$

Boucle : Supprimer $r \in Q$ | Pour $q,p \in Q^2$, $E^{k+1}_{qp} := E^k_{qp} + E^k_{qr}(E^k_{rr})^* E^k_{rp}$. Idem pour E^{k+1}_{ip}, E^{k+1}_{qf}. $\square$

Exemple : Soit l'automate $A_0 =$ 

On élimine (1) :



On élimine (2) : 

Algo 3 : $E \to \mathcal{A}$ en connaissant le Théorème : Méthode des résiduels.

Exemple : On note $L = (a+b)^* a (b^*a + \varepsilon)$. Alors $L_a = L + (b^*a + \varepsilon) = L + \varepsilon$ car $b^*a \subset L$!

$L_b = L$; $L_{aa} = L_a + \varepsilon = L_a$; $L_{ab} = L + b^*a = L$.

D'où l'automate : 

Remarque 3 : La capacité de l'utilisateur à trouver l'automate minimal dépend directement de sa faculté à reconnaître des expressions rationnelles équivalentes...

Correction : C'est une utilisation directe du Théorème de Myhill-Nerode...

Algo 4 $\mathcal{A} \to E$ connaissant le Théorème : Elimination de Gauss par le Lemme d'Arden.

Pour $q \in Q$, on pose $L_q = \sum a L_p \,/\, q \xrightarrow{a} p$ existe dans $\mathcal{A}$, $+\varepsilon$ si $q \in F$. Alors $L_q = \mathcal{L}(\mathcal{A}_q)$.

$w \in L_q \iff \exists p \in \delta^*(q,w), \varepsilon \in L_p \iff \exists p \in \delta^*(q,w), p \in F \iff w \in \mathcal{L}(\mathcal{A}_q)$. $\square$.

Exemple : On reprend l'automate ci-dessus.

$$\begin{cases} L_0 = (a+b)L_1 + aL_2 + aL_3 \\ L_1 = (a+b)L_1 + aL_2 + aL_3 \\ L_2 = bL_2 + aL_3 \\ L_3 = \varepsilon \end{cases} \Rightarrow \begin{cases} L_0 = (a+b)L_1 + aL_2 + a \\ L_1 = L_0 \\ L_2 = b^*a \\ L_3 = \varepsilon \end{cases} \Rightarrow \begin{cases} L_0 = (a+b)^*(ab^*a + a) = \mathcal{L}(\mathcal{A}) \\ L_1 = L_0 \\ L_2 = b^*a \\ L_3 = \varepsilon. \end{cases}$$

Langage des sous-mots

Théorème de Higman

Théorème 1

Soit $L \subset A^*$ et $<$ la relation d'ordre partiel des sous-mots sur A^*.
Soit
- $\underline{L} := \{u \in L, \forall v \in L, v < u \Rightarrow v = u\}$ éléments minimaux de L
- ${}^{>}L := \{u \in A^*, \exists v \in L, v < u\}$ langage des sur-mots de L
- ${}^{<}L := \{u \in A^*, \exists v \in L, u < v\}$ langage des sous-mots de L

Alors
- $\underline{L}$ est fini
- ${}^{>}L$ est rationnel
- ${}^{<}L$ est rationnel

Théorème de Higman

Soit Σ alphabet et L antichaîne de $(\Sigma^*, <)$: deux éléments de L sont comparables ssi ils sont égaux. Alors L est fini.

Higman $\Rightarrow$ Thm1

□ • $\underline{L}$ est une antichaîne, donc fini. ⊢

\+ Tout mot de L est sur-mot d'un élément minimal. Tout sur-mot de L aussi.
Ainsi, ${}^{>}L = \bigcup_{w \in \underline{L}} {}^{>}\{w\}$ et ${}^{>}\{w\} = A^* w_1 \dots A^* w_k A^*$ pour $w = w_1 \dots w_k$.
${}^{>}L$ est une union finie de langages rationnels, il est rationnel. ⊢

⊣ Soit $K := A^* \setminus {}^{<}L$. Mq $K = {}^{>}K$. Alors ${}^{<}L$ est rationnel ▷

◁ Soit $u \in {}^{>}K$ et $v \in K$ tq $v < u$. Mq $u \in K$. Alors, $K = {}^{>}K$, l'inclusion $K \subset {}^{>}K$ étant immédiate.

◁ Si u est sous-mot de L, alors v aussi. Ainsi, $v \in K$ et $v < u$ implique $u \in K$ □

Preuve Thm Higman : Absurde

□ Soit L antichaîne infinie sur Σ. Soit $n := \min_{u \in L} |u|$. On peut choisir L telle que $|\Sigma|$ soit minimal, puis telle que n soit minimal. Alors $|\Sigma| > 1$ car $<$ est totale sur $\{a\}^*$, et $n > 1$ car sinon $L \setminus \{a\}$ pour $a \in L \cap \Sigma$ est une antichaîne infinie sur $\Sigma \setminus \{a\}$.

Trouver $u, v \in L$ tq $u < v$ et $u \neq v$. ▷

◁ Soit $u = u_1 \dots u_n \in L$ de longueur minimal. Soit $L' := \{v \in L, u_1 \dots u_{n-1} < v\}$.
Alors $L \setminus L' \cup \{u_1 \dots u_{n-1}\}$ est une antichaîne contenant un mot de longueur $< n$: elle est finie.
Ainsi, L' est infini, on l'écrit $L' =: (v_i)_{i \in \mathbb{N}}$.
Soit $v_i \in L'$, on peut l'écrire $v_i =: z_{i,1} u_1 \dots z_{i,n-1} u_{n-1} z_{i,n}$ avec $z_{i,j} \in (\Sigma \setminus \{u_j\})^*$ pour $j \in [\![1, n-1]\!]$. Alors $u \not< v_i$ implique $z_{i,n} \in (\Sigma \setminus \{u_n\})^*$.

Soit $Z_j := \{ z_{i,j} \,/\, i \in \mathbb{N} \} \subset (\Sigma \setminus \{u_j\}^*)$ Par minimalité de $|\Sigma|$, l'ensemble des (éventuels) éléments maximaux de Z_j, qui est une antichaîne, est fini.

Mq en faisant n extractions sur $L' = (v_i)_{i\in\mathbb{N}}$, on peut se ramener à $v_1 < v_2$ et $v_1 \neq v_2$ $\triangleright$

$\triangleleft$ Si Z_j est fini, un de ses éléments a une infinité d'antécédents par $v_i \mapsto z_{i,j}$. par extraction, on se ramène au cas où $z_{i,j}$ est constant sur L'

Tous les Z_j ne peuvent être finis, car L' s'injecte dans $\prod_{j=1}^{n} Z_j$.

Soit $j \in [\![1, n]\!]$ tq Z_j est infini. Une extraction permet de supprimer les doublons dans Z_j. Soit $Z_j = (z_{i,j})_{i\in\mathbb{N}}$ Mq on peut en extraire une sous-suite strictement croissante. Alors l'unicité de la décomposition $v_i = z_{i,1} u_1 \dots z_{i,n-1} u_{n-1} z_{i,n}$ et le fait que chaque $(z_{i,j})_{i\in\mathbb{N}}$ soit croissante, avec au moins un $j \in [\![1,n]\!]$ pour lequel $(z_{i,j})$ est strictement croissante signifie que L' est strictement croissante: non seulement v_1 et v_2 conviennent, mais tout couple (v_i, v_j) avec $i < j$ convient $\triangleright$

$\triangleleft$ Soit $N := \max\{|u|,\ u \text{ maximal dans } Z_j\}$ (ou $N = 0$ si Z_j n'a pas d'élément maximal)

Soit $i_1 \in \mathbb{N}$ tq $|z_{i_1,j}| > N$.

Soit $i_2 \in \mathbb{N}$ tq $z_{i_1,j} < z_{i_2,j}$ et $|z_{i_2,j}| > |z_{i_1,j}|$ (⚠ A priori, on n'a pas $i_2 > i_1$)

En itérant le procédé, on construit une suite strictement croissante $(z_{i_k,j})_{k\in\mathbb{N}}$.

L'application $\begin{array}{l} \mathbb{N} \to \mathbb{N} \\ k \mapsto i_k \end{array}$ est injective, on en extrait une application strictement croissante.

Alors la suite $(z_{i_k,j})_{k\in\mathbb{N}}$ est une sous-suite de Z_j strictement croissante $\square$