

I – Régulier $\implies$ reconnaissable par algorithme

I.1 – Expressions régulières linéaires

Définition 1 : Expressions régulières linéaires

Une expression régulière est *linéaire* si chaque lettre y apparaît au plus une fois.

Exemple : ba | a n'est pas linéaire, mais a^*b est linéaire.

Définition 2 : Linéarisation

Linéariser une expression e consiste à remplacer chaque occurrence de lettre a par une nouvelle lettre $\varphi(a)$ différente.

Exemple : $e = \varepsilon$ | $b(a$ | $bb)^*b$ peut se linéariser en $e' = \varepsilon$ | $b_0(a_0$ | $b_1b_2)^*b_3$.

I.2 – Langages locaux

Définition 3 : $P(L)$, $S(L)$, $F(L)$, $N(L)$

Pour un langage $L \subset \Sigma^*$:

- $P(L) = \{a \in \Sigma \mid a\Sigma^* \cap L \neq \emptyset\}$: premières lettres.
- $S(L) = \{a \in \Sigma \mid \Sigma^*a \cap L \neq \emptyset\}$: dernières lettres.
- $F(L) = \{m \in \Sigma^2 \mid \Sigma^*m\Sigma^* \cap L \neq \emptyset\}$: facteurs de taille 2.
- $N(L) = \Sigma^2 \setminus F(L)$: facteurs de taille 2 interdits.

Exercice 1

Donner ces ensembles pour $L = a(a \mid b)^*$.

Définition 4 : Langage local

Soit $\text{Loc}(L)$ l'ensemble des mots dont la première lettre appartient à $P(L)$, la dernière à $S(L)$ et sans facteur de $N(L)$:

$$\text{Loc}(L) = (P(L)\Sigma^* \cap \Sigma^*S(L)) \setminus \Sigma^*N(L)\Sigma^*$$

On dit que L est *local* si $\text{Loc}(L) = L \setminus \{\varepsilon\}$, c'est-à-dire, comme $L \setminus \{\varepsilon\} \subset L$ est toujours vrai :

$$\underbrace{m_1 \in P(L) \wedge m_n \in S(L) \wedge (\forall k \in \llbracket 1, n-1 \rrbracket, m_k m_{k+1} \in F(L))}_{m \in \text{Loc}(L)} \implies m \in L$$

On note $\mathcal{L}_{\text{loc}}(\Sigma)$ l'ensemble des langages locaux sur Σ .

Exercice 2

Montrer que $\mathcal{L}_{\text{loc}}(\Sigma) \subset \mathcal{L}_{\text{reg}}(\Sigma)$.

Exercice 3

Les langages suivants sont-ils locaux ?

1. $L_1 = a^*$
2. $L_2 = (ab)^*$
3. $L_3 = a^* \mid (ab)^*$
4. $L_4 = a^*(ab)^*$

Théorème 1

- Si $L_1 \in \mathcal{L}_{\text{loc}}(\Sigma_1)$, $L_2 \in \mathcal{L}_{\text{loc}}(\Sigma_2)$ avec $\Sigma_1 \cap \Sigma_2 = \emptyset$ alors $L_1 \cup L_2 \in \mathcal{L}_{\text{loc}}(\Sigma_1 \cup \Sigma_2)$ et $L_1 L_2 \in \mathcal{L}_{\text{loc}}(\Sigma_1 \cup \Sigma_2)$.
- Si $L \in \mathcal{L}_{\text{loc}}(\Sigma)$ alors $L^* \in \mathcal{L}_{\text{loc}}(\Sigma)$.

Exercice 4

1. La stabilité par union et concaténation reste-t-elle vraie si $\Sigma_1 \cap \Sigma_2 \neq \emptyset$?
2. $\mathcal{L}_{\text{loc}}(\Sigma)$ est-il stable par complémentaire ?

Théorème 2 : Linéaire $\implies$ local

Si e est une expression régulière linéaire alors $\mathcal{L}(e)$ est local.

I.3 – Automate local

Définition 5 : Automate local

Un AD $A = (\Sigma, Q, q_0, F, \delta)$ est *local* si toutes les transitions étiquetées par une même lettre aboutissent au même état :

$$\forall q_1, q_2 \in Q, a \in \Sigma, \delta(q_1, a) = \delta(q_2, a)$$

Théorème 3

$L \in \mathcal{L}_{\text{loc}}(\Sigma) \iff L$ est reconnu par un automate local.

Exercice 5

Construire un automate local reconnaissant $a(a \mid b)^*$.

I.4 – Algorithme de Berry-Sethi

Algorithme 1 : Berry-Sethi

Entrée : une expression régulière e .

Sortie : un automate de *Glushkov* reconnaissant $\mathcal{L}(e)$.

Linéariser e en e' .

Calculer $P(\mathcal{L}(e'))$, $S(\mathcal{L}(e'))$ et $F(\mathcal{L}(e'))$, et tester $\varepsilon \in \mathcal{L}(e')$.

Construire l'automate local A' reconnaissant $\mathcal{L}(e')$ comme dans la preuve de [Théorème 3](#).

Renvoyer l'automate A obtenu en remplaçant chaque étiquette a' de A' par sa lettre d'origine $\pi(a')$.

Prolongeons π aux mots avec $\pi(m) = \pi(m_1)\dots\pi(m_n)$ pour tout mot $m = m_1\dots m_n$. Alors $\pi(L_1 \cup L_2) = \pi(L_1) \cup \pi(L_2)$, $\pi(L_1 L_2) = \pi(L_1)\pi(L_2)$ et $\pi(L_1^*) = \pi(L_1)^*$. Une induction sur e donne donc $\pi(\mathcal{L}(e')) = \mathcal{L}(e)$.

Remplacer a' par $\pi(a')$ conserve les chemins et leurs extrémités : un chemin acceptant m dans A' accepte $\pi(m)$ dans A , et inversement. Ainsi :

$$\mathcal{L}(A) = \pi(\mathcal{L}(A')) = \pi(\mathcal{L}(e')) = \mathcal{L}(e).$$

Exercice 6

Appliquer [Algorithme : Berry-Sethi](#) à $e = a(a \mid b)^*$.

I.5 – ε -transitions

Définition 6 : Automate avec ε -transitions

Un automate avec ε -transitions permet des transitions étiquetées par ε : $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q)$.

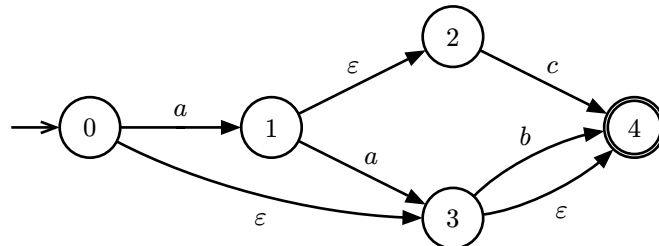


Fig. 1. – Automate à ε -transitions reconnaissant $\{\varepsilon, b, ac, aa, aab\}$.
 $\varepsilon(0) = \{0, 3, 4\}$ et $\varepsilon(1) = \{1, 2\}$

Définition 7 : ε -fermeture

Pour un ensemble d'états $X \subset Q$, on note $\varepsilon(X)$ l' ε -fermeture de X , c'est-à-dire l'ensemble des états atteignables depuis X en utilisant uniquement des ε -transitions.

Exercice 7

Proposer deux algorithmes pour calculer tous les $\varepsilon(q)$ pour $q \in Q$. Comparer.

Théorème 4 : Suppression des ε -transitions

Tout automate avec ε -transitions est équivalent à un automate sans ε -transition.

Exercice 8

Donner un automate sans ε -transition équivalent à Fig. 1.

Remarque : un automate avec ε -transitions n'est pas déterministe, mais il peut être déterminisé en adaptant l'algorithme de déterminisation ou en utilisant préalablement [Théorème : Suppression des \$\varepsilon\$ -transitions](#).

HP I.6 – Automate de Thompson

La construction de Thompson est une alternative récursive à [Algorithme : Berry-Sethi](#) pour obtenir un automate à partir d'une expression régulière.

Exercice 9 : Algorithme de Thompson

1. Définir récursivement un automate $T(e)$ reconnaissant une expression régulière e .
2. Dessiner $T(a(ab \mid b)^*)$.
3. Comparer les tailles des automates de Thompson et de Glushkov.

II – Reconnaisable $\implies$ régulier

Lemme 1 : Normalisation

Tout automate est équivalent à un automate avec ε -transitions *normalisé*, c'est-à-dire avec :

- Un unique état initial sans transition entrante.
- Un unique état final sans transition sortante.

Définition 8 : Automate généralisé

Un *automate généralisé* $A = (\mathcal{R}(\Sigma), Q, I, F, T)$ sur un alphabet Σ permet des transitions étiquetées par des expressions régulières avec $T \subset Q \times \mathcal{R}(\Sigma) \times Q$ fini.

Un mot m est *accepté* par A s'il existe un *chemin acceptant* de I à F $q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \dots \xrightarrow{e_n} q_n$ tel que $m \in \mathcal{L}(e_1) \dots \mathcal{L}(e_n)$.
Le *langage* $\mathcal{L}(A)$ *reconnu* par A est l'ensemble des mots acceptés par A .

Remarques :

- Un automate avec ε -transitions est un cas particulier d'automate généralisé.
- Une transition de $p \xrightarrow{e} q$ peut être franchie en lisant un mot quelconque de $\mathcal{L}(e)$.

Théorème 5

Tout automate généralisé est équivalent à un automate fini classique.

Exercice 10 : Substitution régulière

Soit $L \subset \Sigma^*$ un langage reconnaissable, Γ un alphabet et $s : \Sigma \rightarrow \mathcal{R}(\Gamma)$ qu'on étend à $s : \Sigma^* \rightarrow \mathcal{R}(\Gamma)$ par :

$$s(m_1 \dots m_n) = s(m_1) \dots s(m_n), \quad s(\varepsilon) = \varepsilon$$

Montrer que $s(L)$ est reconnaissable.

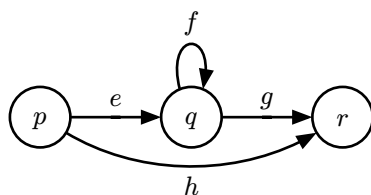
Lemme 2 : Élimination d'un état

Soit $A = (\mathcal{R}(\Sigma), Q, I, F, \delta)$ un automate généralisé et $q \in Q \setminus (I \cup F)$.

Alors A est équivalent à l'automate $A' = (\mathcal{R}(\Sigma), Q \setminus \{q\}, I, F, \delta')$ où, pour tout $p, r \in Q \setminus \{q\}$:

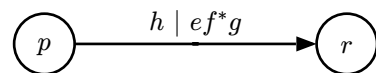
$$\delta'(p, r) = \delta(p, r) \mid \delta(p, q)\delta(q, q)^*\delta(q, r)$$

Remarque : $\delta(p, q) = \emptyset$ si p n'a pas de transition vers q , et $\delta(q, q)^* = \varepsilon$ si q n'a pas de boucle.



Automate A

$\longrightarrow$



Automate A' après élimination de q

Algorithme 2 : Élimination des états

Entrée : un automate normalisé $A = (\Sigma, Q, q_i, q_f, \delta)$.

Sortie : une expression régulière e telle que $\mathcal{L}(e) = \mathcal{L}(A)$.

Tant que $|Q| > 2$:

 Choisir un état $q \in Q \setminus \{q_i, q_f\}$.

 Éliminer q comme dans [Lemme : Élimination d'un état](#).

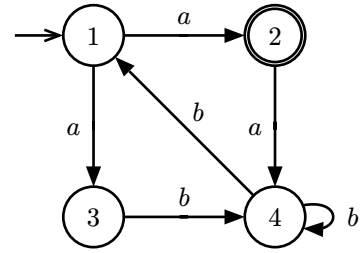
Renvoyer l'étiquette de q_i à q_f , ou $\emptyset$ si pas de transition.

Variant : $|Q|$ diminue à chaque itération.

Invariant : l'automate reconnaît toujours $\mathcal{L}(A)$.

Exercice 11

Appliquer l'algorithme d'élimination des états.



Exercice 12 : Analyse de [Algorithme : Élimination des états](#)

1. Quelle structure de données utiliser dans [Algorithme : Élimination des états](#) ?
2. Donner alors les complexités temporelle et spatiale, en fonction de $n = |Q|$.
3. Montrer que l'expression régulière renvoyée peut être de taille exponentielle en n . Est-ce en contradiction avec la question précédente ?

III – Théorème de Kleene

Théorème 6 : Théorème de Kleene

Un langage est régulier si et seulement s'il est reconnaissable par un automate fini : $\mathcal{L}_{\text{rec}}(\Sigma) = \mathcal{L}_{\text{reg}}(\Sigma)$.

Corollaire 1 : Stabilités de $\mathcal{L}_{\text{rec}}$

Les langages réguliers sont stables par union finie, concaténation, étoile, intersection finie, complémentaire et différence.

Attention : $\mathcal{L}_{\text{rec}}$ n'est pas stable par inclusion, ni union ou intersection infinie.