Chapitre 1

Langage Automate

Hugo Salou MPI*

Dernière mise à jour le 21 septembre 2022

1 Motivation

1.1 1ère motivation

Il y a une grande différence entre les mathématiques et l'informatique : la gestion de l'infini. On n'a pas de mémoire infinie sur un ordinateur. Par exemple, pour représenter π ou $\sqrt{2}$, on ne peut pas stocker un nombre infini de décimales. Ce n'est pas une question de base, ces nombres ont aussi des décimales infinies dans une base 2.

Par exemple, si on ne veut utiliser $\sqrt{2}$ seulement pour plus tard le mettre au carré. On peut définir une structure en `C` comme celle qui suit

```
1 typedef struct {
2   int carre;
3   bool sign;
4 };
```

Code 1 – $\sqrt{2}$ sous forme de structure

On a pu décrire $\sqrt{2}$ comme cela car il y a une certaine régularité dans ce nombre.

On définit des relations entre ces objets. Dans ce chapitre, on va commencer par étudier autre chose : les mots. Les mots sont utilisés, tout d'abord, pour entrer une liste de lettres mais aussi le programme lui-même. En effet, il y a une liste infinie de code possibles en `C`.

1.2 2nde motivation

Les ordinateurs sont complexes ; il peut être dans une multitude d'états. On représente une succession de tâches (le symbole ● représente une tâche) :

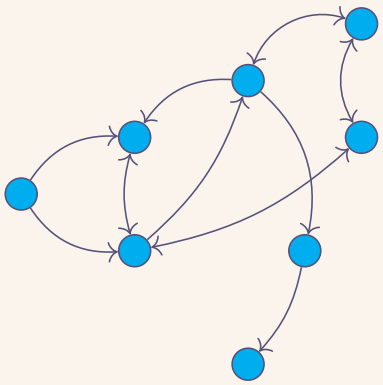


Figure 1 – États d'un ordinateur

Par exemple, pour une boucle infinie, on a un cycle dans le graphe ci-dessus. Ou, s'il atteint un certain nœud, on a un *bug*.

Mais, pour résoudre ce problème, on peut forcer le nombre d'état d'un ordinateur ; par exemple, dire qu'un ordinateur a 17 états.

On décide donc de représenter mathématiquement un ordinateur afin de pouvoir faire des preuves avec. Et, c'est l'objet de ce chapitre.

2 Mots et languages, rappels

Définition: On appelle *alphabet* un ensemble fini, d'éléments qu'on appelle *lettres*.

Définition: On appelle une *mot* sur Σ (où Σ est un alphabet) une suite finie de lettres de Σ.

La *longueur* d'un mot est le nombre de lettres, comptées avec leurs multiplicité. On la note $|\omega|$ pour un mot ω. Si $|\omega| = n \in \mathbb{N}^*$, on indexe les lettres de ω pour $(\omega_i)_{i\in[\![1,n]\!]}$ et on écrit alors

$$\omega = \omega_1\omega_2\omega_3\dots\omega_n.$$

Il existe un unique mot de longueur 0 appelé *mot vide*, on le note ε.

Définition: Si Σ est un alphabet, on note
- Σ^n les mots de longueur n ;
- Σ^* les mots de longueurs positives ou nulle ;
- Σ^+ les mots de longueurs strictement positives.

Remarque:

$$\Sigma^* = \bigcup_{n\in\mathbb{N}} \Sigma^n \qquad \Sigma^+ = \bigcup_{n\in\mathbb{N}^*} \Sigma^n.$$

Définition: Soit Σ un alphabet. Soit x et y deux mots de Σ^*. Notons

$$x = x_1x_2x_3\dots x_n$$
$$y = y_1y_2y_3\dots y_n.$$

On définit alors *concaténation* notée $x \cdot y$ l'opération définie comme

$$x \cdot y = x_1x_2x_3\dots x_ny_1y_2\dots y_n.$$

Remarque:
- $\cdot$ est une opération interne ;
- ε est neutre pour $\cdot$: $\forall n \in \Sigma^*, \varepsilon \cdot x = x \cdot \varepsilon = x$.
- $\cdot$ est associatif.

(On dit que c'est un monoïde.)

Remarque:
L'opération $\cdot$ n'est pas commutative.

Définition: Soit x et y deux mots sur l'alphabet Σ. On dit que
- x est un *préfixe* de y si $\exists v \in \Sigma^*, y = x \cdot v$;
- x est un *suffixe* de y si $\exists v \in \Sigma^*, y = v \cdot x$;
- x est un *facteur* de y si $\exists (v, w) \in \left(\Sigma^*\right)^2, y = v \cdot x \cdot w$.

Exemple:
- a est facteur de aa ;
- ε est un facteur de a.

Définition: On dit que x est un *sous-mot* de y si x est une suite extraite de y. Par exemple

$$a \not{b} a\, a \not{b} a \qquad \longrightarrow \qquad a\, a\, a\, a.$$

Définition: Un *langage* est un ensemble de mots. C'est donc un élément de $\mathscr{P}(\Sigma^*)$.

Remarque (⚠):
$\varnothing \neq \{\varepsilon\}$.

3 Language régulier

3.1 Opérations sur les langages

Remarque:
Les langages sont des ensembles. On peut donc leurs appliquer des opérations ensemblistes.

Définition: Soient $L_1, L_2 \in \mathscr{P}(\Sigma^*)$ deux langages. On définit la *concaténation* de deux langages, notée $L_1 \cdot L_2$:

$$L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1,\, v \in L_2\}.$$

Remarque: — L'opération $\cdot$ (languages) a $\{\varepsilon\}$ pour neutre.
— L'opération $\cdot$ (languages) a $\varnothing$ pour élément absorbant :

$$\forall L \in \mathscr{P}(\Sigma^*),\ L \cdot \varnothing = \varnothing \cdot L = \varnothing.$$

— L'opération $\cdot$ est distributive : soient K, L, M trois langages ; on a

$$K \cdot (L \cup M) = (K \cdot L) \cup (K \cdot M);$$
$$(L \cup M) \cdot K = (L \cdot K) \cup (M \cdot K).$$

Définition: Étant donné un langage L, on définit par récurrence :
— $L^0 = \{\varepsilon\}$;
— $L^{n+1} = L^n \cdot L = L \cdot L^n$. [1]
On note alors

$$L^* = \bigcup_{n \in \mathbb{N}} L^n \qquad \text{et} \qquad L^+ = \bigcup_{n \in \mathbb{N}} L^n.$$

Remarque:
On a $\Sigma^* = \Sigma^{*^*}$. On notera donc Σ^* dans tous les cas.

Remarque:
Si $\varepsilon \in L$, alors $L^* = L^+$. En effet, $L^* = L^+ \cup \{\varepsilon\}$.

Remarque:
On nomme l'application $L \mapsto L^*$ l'*étoile de* Klein.

Remarque:
Avec L et K deux alphabets, on a

$$|L \cdot K| \leqslant |L|\,|K|.$$

En effet, avec $K = \{a, aa\}$, on a $K^2 = \{aa, aaa, aaaa\}$.

Exemple:
Avec $L = \{a, ab\}$, on a
— $L^1 = L$;
— $L^2 = \{aa,\ aab,\ aba\,abab\}$;
— $L^* \supseteq \{\varepsilon,\ a,\ ab,\ aa, \ldots, aaaaaaa \ldots\}$.

1. La deuxième égalité est assurée par l'associativité de l'opération $\cdot$.

Définition: Soit Σ un alphabet. On appelle *ensemble des langages réguliers*, noté LR. Le plus petit ensemble tel que
- $\varnothing \subset \mathrm{LR}$;
- Si $a \in \Sigma$, alors $a \in \mathrm{LR}$;
- Si $L_1 \in \mathrm{LR}$ et $L_2 \in \mathrm{LR}$, on a $L_1 \cup L_2 \in \mathrm{LR}$;
- Si $L_1 \in \mathrm{LR}$ et $L_2 \in \mathrm{LR}$, on a $L_1 \cdot L_2 \in \mathrm{LR}$;
- Si $L \in \mathrm{LR}$, alors $L^* \in \mathrm{LR}$.

Exemple:
- Soit $\Sigma = \{t, o\}$. Est-ce que $\{toto\} \in \mathrm{LR}$? On sait déjà que $\{t\}$ et $\{o\}$ sont déjà des langages réguliers. Or,
$$\{toto\} = \{t\} \cdot \{o\} \cdot \{t\} \cdot \{o\}.$$
Et donc $\{toto\} \in \mathrm{LR}$.
- On a $\{\varepsilon\} = \varnothing^* \in \mathrm{LR}$.
- On a
$$\left\{ \begin{array}{cccc} x_{11} & x_{12} & \dots & x_{1,n_1} \\ x_{21} & x_{22} & \dots & x_{2,n_2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n_m} \end{array} \right\} \in \mathrm{LR}$$
car $\{x_{i,1}, x_{i,2}, \dots, x_{i,n_i}\} \in \mathrm{LR}$ et c'est stable par union finie.
- Avec $\Sigma = \{a, b\}$, on a
$$L = \{\varepsilon,\, a,\, aa,\, aaa, \dots\} = \underbrace{\left(\overbrace{\{a\}}^{\in \mathrm{LR}}\right)^*}_{\in \mathrm{LR}}.$$
- $\Sigma^* \in \mathrm{LR}$.
- Contre-exemple : $\{a^n \mid a \text{ premier}\} \notin \mathrm{LR}$.

3.2 Expressions régulières

On représente $\varnothing$ l'ensemble vide, $\_|\_$ l'union de deux expressions régulières, $\_ \cdot \_$ la concaténation de deux expression régulières, et, $\_^*$ l'étoile de Klein pour les expressions régulières.

On cherche à représenter informatiquement ces expressions à l'aide d'une règle de construction nommée.

Définition: Étant donné un alphabet Σ, on définit $\mathrm{Reg}(\Sigma)$ défini par induction nommée à partir des règles :

- $\varnothing\big|^0$;
- $|\,\big|^2$;
- $\cdot\,\big|^2$;
- $*\,\big|^1$;
- $L\big|^1_\Sigma$;
- $\varepsilon\big|^0$.

Ces règles peuvent être définies en OCamL (douteux) de la façon suivante :

```
1 type regex =
2   | ∅
3   | |  of regex * regex
4   | ·  of regex * regex
5   | *  of regex
6   | L of char
7   | ε
```

Code 2 – Règles des expressions régulières en OCamL

On peut donc écrire

$$\big((\varnothing^*) \mid (a \cdot b)\big) \quad \longrightarrow \quad |(*(\varnothing()), \cdot(L(a), L(b))).$$

A-t-on $|(L(a), L(b)) \stackrel{?}{=} |(L(b), L(a))$? Non, sinon on risque d'avoir une boucle infinie au moment de l'évaluation de cette expression.

On peut simplifier la notation : au lieu d'écrire $|(*(\varnothing()), \cdot(L(a), L(b)))$, on note $\varnothing^* \mid (a \cdot b)$.

Définition: On définit

$$\begin{aligned}
\mathscr{L} : \mathrm{Reg}(\Sigma) &\longrightarrow \mathscr{P}(\Sigma^*) \\
\varnothing &\longmapsto \varnothing \\
a &\longmapsto \{a\} \\
\varepsilon &\longmapsto \{\varepsilon\} \\
e_1 \cdot e_2 &\longmapsto \mathscr{L}(e_1) \cdot \mathscr{L}(e_2) \\
e_1 \mid e_2 &\longmapsto \mathscr{L}(e_1) \cup \mathscr{L}(e_2) \\
e^* &\longmapsto \mathscr{L}(e)^*
\end{aligned}$$

Exemple:
Deux expressions régulières peuvent donner le même language : on a $\mathscr{L}(\varnothing) = \varnothing$ mais également $\mathscr{L}(\varnothing \mid \varnothing) = \mathscr{L}(\varnothing) \cup \mathscr{L}(\varnothing) = \varnothing \cup \varnothing = \varnothing$. De même, $\mathscr{L}\big(a \mid (b \cdot b^*)\big) = \{a, b, bb, bbb, \ldots\} = \mathscr{L}\big((bb^*) \mid a\big)$.

Définition: On définit sur $\mathrm{Reg}(\Sigma)$ la fonction "vars" définie comme

$$\begin{aligned}
\mathrm{vars} : \mathrm{Reg}(\Sigma) &\longrightarrow \mathscr{P}(\Sigma) \\
\varnothing &\longmapsto \varnothing \\
\varepsilon &\longmapsto \varnothing \\
a \in \Sigma &\longmapsto \{a\} \\
e_1 \cdot e_2 &\longmapsto \mathrm{vars}(e_1) \cup \mathrm{vars}(e_2) \\
e_1 \mid e_2 &\longmapsto \mathrm{vars}(e_1) \cup \mathrm{vars}(e_2) \\
e^* &\longmapsto \mathrm{vars}(e).
\end{aligned}$$

Propriété: Un langage L est régulier si et seulement s'il existe $e \in \mathrm{Reg}(\Sigma)$ telle que $\mathscr{L}(e) = L$.

Preuve: " $\Longleftarrow$ " Montrons que, pour toute expression régulière $e \in \mathrm{Reg}(\Sigma)$, P_e : "le langage $\mathscr{L}(e)$ est régulier." On procède par induction.
- $P_\varnothing : \mathscr{L}(\varnothing) = \varnothing \in \mathrm{LR}$;
- $P_\varepsilon : \mathscr{L}(\varepsilon) = \{\varepsilon\} = \varnothing^* \in \mathrm{LR}$;
- Soit $a \in \Sigma$. Montrons $P_a : \mathscr{L}(a) = \{a\} \in \mathrm{LR}$;
- Soient e_1 et e_2 deux expressions régulières telles que P_{e_1} et P_{e_2} soient vrais. Montrons $P_{e_1 \cdot e_2} : \mathscr{L}(e_1 \cdot e_2) = \underset{\in \mathrm{LR}}{\mathscr{L}(e_1)} \cdot \underset{\in \mathrm{LR}}{\mathscr{L}(e_2)} \in \mathrm{LR}$
- Soient e_1 et e_2 deux expressions régulières telles que P_{e_1} et P_{e_2} soient vrais. Montrons $P_{e_1 | e_2} : \mathscr{L}(e_1 \mid e_2) = \underset{\in \mathrm{LR}}{\mathscr{L}(e_1)} \cup \underset{\in \mathrm{LR}}{\mathscr{L}(e_2)} \in \mathrm{LR}$

" $\Longrightarrow$ " Montrons que, pour tout langage régulier L, il existe une expression régulière e de l'alphabet Σ telle que $\mathscr{L}(e) = L$. Soit X l'ensemble des langages L tels qu'il

existent une expression régulière e de l'alphabet Σ telle que $\mathscr{L}(e) = L$. On a

$$X \supseteq \underset{\mathscr{L}(\varnothing)}{\{\varnothing\}} \cup \big\{\underset{\mathscr{L}(a)}{\{a\}} \mid a \in \Sigma\big\}.$$

De plus, si deux langages L_1 et L_2 sont dans X, alors il existent e_1 et e_2 deux expressions régulières de Σ telle que $\mathscr{L}(e_1) = L_1$ et $\mathscr{L}(e_2) = L_2$. Or, $\mathscr{L}(e_1 \mid e_2) = \mathscr{L}(e_1) \cup \mathscr{L}(e_2) = L_1 \cup L_2$ et donc $L_1 \cup L_2$.
De même pour $L_1 \cdot L_2$.
Si un langage L est dans X, alors il existe une expression régulière e d'un alphabet Σ telle que $\mathscr{L}(e) = L$, alors $\mathscr{L}(e^*) = L^* \in X$.
X contient les langages $\varnothing$ et $\{a\}$ (avec $a \in \Sigma$) et X est stable par $\cup$, $\cdot$ et $*$. Or, LR est défini comme le plus petit ensemble vérifiant les propriétés et donc LR $\subseteq X$.

□

Remarque (Notation):
Les notations Reg(Σ) et Regexp(Σ) sont équivalentes.

4 Automates finis (sans ε-transitions)

On considère l'automate représenté par les états suivants. L'entrée est représentée par la flèche sans nœud de départ et la sortie par celle sans nœud d'arrivée.

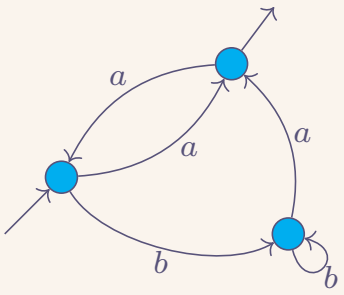


Figure 2 – Exemple d'automate

On représente une séquence d'état par un mot comme *aaba* (qui correspond à une séquence valide) ou *bbab* (qui n'est pas valide).

4.1 Définitions

Définition: Un *automate fini* (sans ε-transition) est un quintuplet $(Q, \Sigma, I, F, \delta)$ où
- Q est un ensemble d'états ;
- Σ est son alphabet de travail ;
- $I \subseteq Q$ est l'ensemble des états initiaux ;
- $F \subseteq Q$ est l'ensemble des états finaux.
- $\delta \subseteq Q \times \Sigma \times Q$.

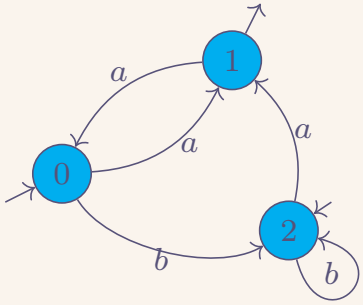


FIGURE 3 – Exemple d'automate (2)

EXEMPLE:
L'automate ci-dessus est donc représenté mathématiquement par

$$\Big(\underbrace{\{0,1,2\}}_{Q},\underbrace{\{a,b\}}_{\Sigma},\underbrace{\{0,2\}}_{I},\underbrace{\{1\}}_{F},\underbrace{\{(0,a,1),(0,b,2),(1,a,0),(2,a,1),(2,b,2)\}}_{\delta}\Big).$$

Définition: On dit qu'une suite $(q_0, q_1, q_2, \dots, q_n) \in Q^{n+1}$ qu'elle est *une suite de transition de l'automate* $\mathscr{A} = (Q, \Sigma, I, F, \delta)$ dès lors qu'il existe $(a_1, a_2, \dots, a_n) \in \Sigma^n$ tels que, pour out $i \in [\![1, n]\!]$, $(q_{i-1}, a_i, q_i) \in \delta$. On note parfois une telle suite de transition par

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} q_3 \to \dots \to q_n - 1 \xrightarrow{a_n} q_n.$$

EXEMPLE:
$1 \xrightarrow{a} 0 \xrightarrow{b} 2 \xrightarrow{b} 2$ est une suite de transitions de l'automate ci-avant.

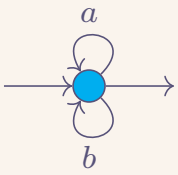
Définition: On dit qu'une suite $(q_0, q_1, \dots, q_n)$ est une *execution* dans l'automate $\mathscr{A} = (Q, \Sigma, I, F, \delta)$ si c'est une suite de transition de A telle que $q_0 \in I$. On dit également qu'elle est *acceptante* si $q_n \in F$.

Lorsque $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \to \dots \to q_{n-1} \xrightarrow{a_n} q_n$ est une suite de transitions d'un automate $\mathscr{A}$, on dit que le mot $a_1 a_2 \dots a_n$ est l'*étiquette* de cette transition.

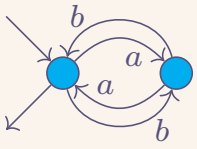
Définition (Langage reconnu par un automate)**:** On dit qu'un mot $\omega \in \Sigma^*$ est *reconnu* par un automate $\mathscr{A} = (Q, \Sigma, I, F, \delta)$ s'il est l'étiquette d'une execution acceptable de $\mathscr{A}$. On note alors $\mathscr{L}(\mathscr{A})$ l'ensemble des mots reconnus par l'automate $\mathscr{A}$.

Définition: On dit d'un langage L qu'il est *reconnaissable* s'il existe un automate $\mathscr{A}$ tel que $\mathscr{L}(\mathscr{A}) = L$. On note $\mathrm{Reg}(\Sigma)$ l'ensemble des mots reconnaissables.

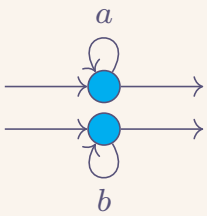
EXEMPLE:
Avec $\Sigma = \{a, b\}$, on cherche $\mathscr{A}$ tel que
— $\mathscr{L}(\mathscr{A}) = \Sigma^*$
— $\mathscr{L}(\mathscr{A}) = (\Sigma^2)^*$



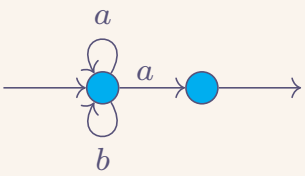
(a) Automate minimal pour $\mathscr{L}(\mathscr{A}) = \Sigma^*$



(b) Automate minimal pour $\mathscr{L}(\mathscr{A}) = (\Sigma^2)^*$



(c) Automate minimal pour $\mathscr{L}(\mathscr{A}) = \{a\}^* \cup \{b\}^*$



(d) Automate minimal pour $\mathscr{L}(\mathscr{A}) = \sigma^* \cdot \{a\}$

FIGURE 4 – Automates minimaux pour différentes valeurs de $\mathscr{L}(\mathscr{A})$

Définition (Automate déterministe)**:** On dit d'un automate $\mathscr{A} = (Q, \Sigma, I, F, \delta)$ qu'il est *déterministe* si

1. $|I| = 1$;
2. $\forall (q, q_1, q_2) \in Q^3$, $\forall a \in \Sigma$, $(q, a, q_1) \in \delta$ et $(q, a, q_2) \in \delta \implies q_1 = q_2$;

REMARQUE:
(2) est équivalent à

$$\forall (q, a) \in Q \times \Sigma, \left|\{q' \in Q \mid (q, a, q') \in \delta\}\right| \leqslant 1.$$

Définition (Automate complet)**:** On dit qu'un automate $\mathscr{A} = (Q, \Sigma, I, F, \delta)$ qu'il est *complet* si

$$\forall (q, a) \in Q \times \Sigma, \exists q' \in Q, (q, a, q') \in \delta.$$

EXEMPLE:
Les automates ci-avant sont

(a) complet et déterministe ;
(b) complet et déterministe ;
(c) non complet et non déterministe ;
(d) non complet et non déterministe.

4.2 Transformations en automates équivalents

On peut représenter le langage utilisé par l'automate ci-dessous avec une expression régulière : $a^* \cdot (a \mid bab)$. L'arbre ci-dessous n'est pas déterministe. On cherche à le rendre déterministe : pour cela, on trace un arbre contenant les nœuds accédés en fonctions de l'expression lue.

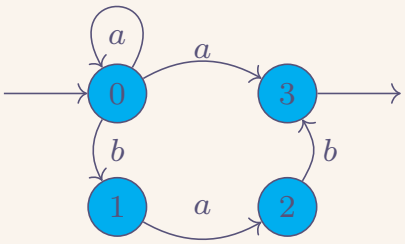


Figure 5 – Automate non déterministe ayant pour expression régulière $a^* \cdot (a \mid bab)$

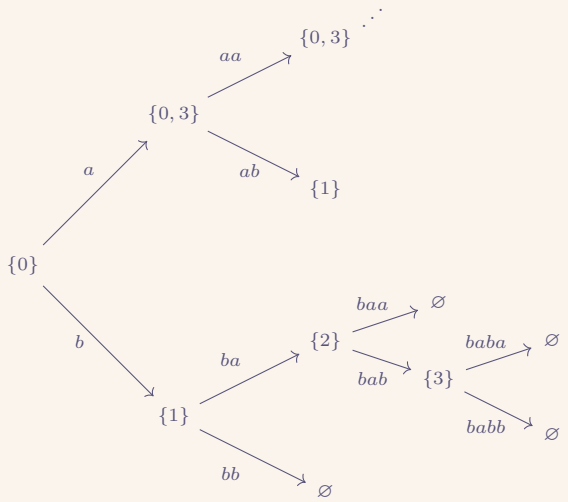


Figure 6 – Nœuds possibles par rapport à l'expression lue

À l'aide de cet arbre, on peut trouver un automate déterministe équivalent à l'automate précédent.

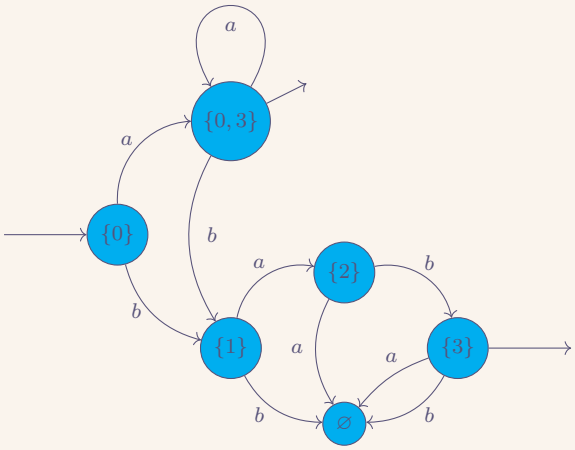


Figure 7 – Automate déterministe ayant pour expression régulière $a^* \cdot (a \mid bab)$

Définition: On dit de deux automates $\mathscr{A}$ et $\mathscr{A}'$ qu'ils sont *équivalents* si $\mathscr{L}(\mathscr{A}) = \mathscr{L}(\mathscr{A}')$.

Théorème: Pour tout automate $\mathscr{A}$, il existe un automate déterministe $\mathscr{A}'$ tel que $\mathscr{L}(\mathscr{A}) = \mathscr{L}(\mathscr{A}')$.

Preuve:
Soit $\mathscr{A} = (\mathcal{Q}, \Sigma, I, F, \delta)$ un automate. On pose $\Sigma' = \Sigma$, $\mathcal{Q}' = \mathscr{P}(\mathcal{Q})$, $I' = \{I\}$, $F = \{Q \in \mathscr{P}(\mathcal{Q}) \mid Q \cap F \neq \varnothing\}$ et

$$\delta' = \left\{ (Q, a, Q') \in \mathcal{Q}' \times \Sigma \times \mathcal{Q}' \;\middle|\; Q' = \left\{ q' \in \mathcal{Q} \mid \exists q \in Q,\, (q, a, q') \in \delta \right\} \right\}.$$

On pose alors l'automate $\mathscr{A}' = (\mathcal{Q}', \Sigma', I', F', \delta')$. Montrons que $\mathscr{L}(\mathscr{A}') = \mathscr{L}(\mathscr{A})$. On procède par double-inclusion.

"$\subseteq$" Soit $w \in \mathscr{L}(\mathscr{A})$. Il existe donc une exécution acceptante $I \ni q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} q_2 \to \cdots \to q_{n-1} \xrightarrow{w_n} q_n \in F$ telle que $w = w_1 w_2 \dots w_n$. On pose $Q_0 = I$, et, pour tout entier $i \in [\![1, n]\!]$,

$$Q_i = \left\{ q' \in \mathcal{Q} \mid \exists q \in Q_{i-1},\, (q, w_i, q') \in \delta \right\}.$$

Remarquons que, pour tout entier $i \in [\![1, n]\!]$, on a $(Q_{i-1}, w_i, Q_i) \in \delta'$. On a donc $I' \ni Q_0 \xrightarrow{w_1} Q_1 \to \cdots \to Q_{n-1} \xrightarrow{w_n} Q_n$ est une exécution de $\mathscr{A}'$. Montrons que, pour tout $i \in [\![0, n]\!]$, on a $q_i \in Q_i$ par récurrence finie.

— $q_0 \in I = Q_0$.

— Soit $p < n$ tel que $q_p \in Q_p$ alors q_{p+1} est tel que $(q_p, w_{p+1}, q_{p+1}) \in \delta$ et q_{p+1} est tel qu'il existe $q \in Q_p$ tel que $(q, w_{p+1}, q_{p+1}) \in \delta$. On en déduit $q_{p+1} \in Q_{p+1}$.

On a donc $q_n \in Q_n$ et $q_n \in F$ donc $Q_n \cap F \neq \varnothing$ et donc $Q_n \in F'$. L'exécution $Q_0 \xrightarrow{w_1} Q_1 \to \cdots \to Q_{n-1} \xrightarrow{w_n} Q_n$ est donc acceptante dans $\mathscr{A}'$ et donc $w = w_1 \dots w_n \in \mathscr{L}(\mathscr{A}')$.

"$\supseteq$" Soit $w \in \mathscr{L}(\mathscr{A}')$. Soit donc $I' \ni Q_0 \xrightarrow{w_1} Q_1 \to \cdots \to Q_{n-1} \xrightarrow{w_n} Q_n \in F'$ une exécution acceptante de w dans $\mathscr{A}'$. $Q_n \cap F \neq \varnothing$. Soit donc $q_n \in Q_n \cap F$. Soit $q_{n-1} \in Q_{n-1}$ tel que $(q_{n-1}, w_n, q_n) \in \delta$ (par définition de $(Q_{n-1}, w_n, Q_n) \in \delta'$). "De proche en proche," il existe $q_0, q_1, \dots, q_{n-2}$ tels que, pour tout $i \in [\![1, n]\!]$, $(q_{i-1}, w_i, q_i) \in \delta$ et $q_i \in Q_i$. Or, $q_0 \in Q_0 \in I' = \{I\}$ donc $Q_0 = I$ et donc $q_0 \in I$. On rappelle que $q_n \in F$. On en déduit donc que $q_0 \xrightarrow{w_1} q_1 \to \cdots \to q_{n-1} \xrightarrow{w_n} q_n$ une exécution acceptante da ns $\mathscr{A}$ et donc $w \in \mathscr{L}(\mathscr{A}')$.

□

Pour comprendre la construction de l'automate dans la preuve, on fait un exemple. On considère l'automate non-déterministe ci-dessous.

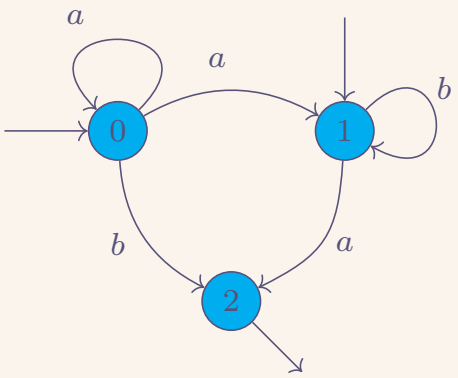


Figure 8 – Automate non déterministe

On construit la *table de transition* :

	a	b
$\varnothing$	$\varnothing$	$\varnothing$
$\{0\}$	$\{0,1\}$	$\{2\}$
$\{1\}$		

Table 1 – Table de transition de l'automate ci-avant

À faire : Finir la table de transition

Remarque:
L'automate $\mathscr{A}'$ construit dans le théorème précédent est complet mais son nombre de nœud suit une exponentielle.

Propriété: Soit $\mathscr{A}$ un automate fini à n états. Il existe un automate $\mathscr{A}'$ ayant $n+1$ états tel que $\mathscr{L}(\mathscr{A}) = \mathscr{L}(\mathscr{A}')$ avec $\mathscr{A}'$ complet.

Preuve:
Soit $\mathscr{A} = (\mathcal{Q}, \Sigma, I, F, \delta)$ un automate à n états. Soit $P \notin \mathcal{Q}$. On pose $\Sigma' = \Sigma$, $\mathcal{Q}' = \mathcal{Q} \cup \{P\}$, $I' = I$, $F' = F$ et

$$\delta' = \delta \cup \left\{ (q, \ell, P) \in \mathcal{Q}' \times \Sigma \times \{P\} \,\middle|\, \forall q' \in \mathcal{Q}',\ (q, \ell, q') \notin \delta \right\}.$$

Montrons que $\mathscr{L}(\mathscr{A}) = \mathscr{L}(\mathscr{A}')$. À faire : Preuve à faire

□

Définition: Soit $\mathscr{A} = (\mathcal{Q}, \Sigma, I, F, \delta)$ un automate. On dit d'un état $q \in \mathcal{Q}$ qu'il est
- *accessible* s'il existe une execution $I \ni q_0 \xrightarrow{w_1} q_1 \to \cdots \to q_{n-1} \xrightarrow{w_n} q_n = q$.
- *co-accessibles*'il existe une suite de transitions $q \xrightarrow{w_1} q_1 \to \cdots \to q_{n-1} \xrightarrow{w_n} q_n \in F$.

Dans l'automate ci-dessous, l'état 0 n'est pas accessible et l'état 2 n'est pas accessible.

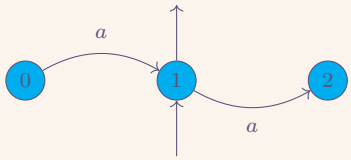


FIGURE 9 – Non-exemples d'états accessibles et co-accessibles

Définition: On dit d'un automate $\mathscr{A}$ qu'il est *émondé* dès lors que chaque état est accessible et co-accessible.

Propriété: Soit $\mathscr{A}$ un automate. Il existe $\mathscr{A}'$ un automate émondé tel que $\mathscr{L}(\mathscr{A}') = \mathscr{L}(\mathscr{A})$.

Preuve:
Soit $\mathscr{A} = (\mathcal{Q}, \Sigma, I, F, \delta)$. On pose $\Sigma' = \Sigma$, $\mathcal{Q}' = \{q \in \mathcal{Q} \mid q \text{ accessible ou co-accessible }\}$, $I' = I \cap R'$, $F' = F \cap R'$ et

$$\delta = \left\{(q, \ell, q') \in \mathcal{Q}' \times \Sigma \times \mathcal{Q}' \;\middle|\; (q, \ell, q') \in \delta\right\} = (\mathcal{Q}', \Sigma, \mathcal{Q}') \cap \delta.$$

Montrons que $\mathscr{L}(\mathscr{A}) = \mathscr{L}(\mathscr{A}')$. On procède par double inclusion. On vérifie aisément que $\mathscr{L}(A') \subseteq \mathscr{L}(\mathscr{A})$. On montre maintenant $\mathscr{L}(\mathscr{A}) \subseteq \mathscr{L}(\mathscr{A}')$. Soit $w \in \mathscr{L}(\mathscr{A}')$. Soit $q_0, \dots, q_n$ tels que $q_0 \xrightarrow{w_1} q_1 \to \cdots \to q_{n-1} \xrightarrow{w_n} q_n$ est une exécution acceptante. Or, pour tout $i \in [\![0, n]\!]$, q_i est accessible et co-accessible donc $q_i \in \mathcal{Q}'$. De plus, $q_0 \in I'$ et $q_n \in F'$. De plus, pour tout $i \in [\![0, n-1]\!]$, $(q_i, w_{i+1}, q_{i+1}) \in \delta'$. Donc, $q_0 \xrightarrow{w_1} q_1 \to \cdots \to q_{n-1} \xrightarrow{w_n} q_n$ est une execution acceptante de $\mathscr{A}'$. □

Parfois, on veut pouvoir "sauter" d'un état à un autre dans un automate. On utilise pour cela des ε-transitions.

5 Automates finis avec ε-transitions

Définition: On dit d'un automate sur l'alphabet $\Sigma \cup \{\varepsilon\}$ que c'est un automate avec ε-transition.

EXEMPLE:
L'automate ci-dessous est un automate avec ε-transitions.

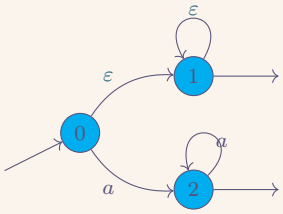


FIGURE 10 – Exemple d'automate avec ε-transition

Définition: Soit $w \in (\Sigma \cup \{\varepsilon\}^*$. On définit alors $\tilde{w}$ le mot obtenu en supprimant les occurrences de ε dans w.

Exemple:
Avec $\Sigma = \{a, b\}$ et $w = ab\varepsilon aa\varepsilon\varepsilon a$, on a $\tilde{w} = abaaa$.

On a également $\tilde{\varepsilon} = \varepsilon$; pour deux mots w_1 et w_2, on a $\widetilde{w_1 \cdot w_2} = \tilde{w}_1 \cdot \tilde{w}_2$; on a également $\tilde{a} = a$ pour $a \in \Sigma$ et $\tilde{\varepsilon} = \varepsilon$.

Définition: Soit $\mathscr{A}$ un automate avec ε-transition. On pose $\tilde{(}\mathscr{A})$ est le langage de l'automate sur l'alphabet $\Sigma \cup \{\varepsilon\}$. On appelle *langage* de A, l'ensemble À faire : retrouver la formule.

Exemple:
On peut trouver un automate reconnaissant la concaténation des langages des deux automates ci-dessous.

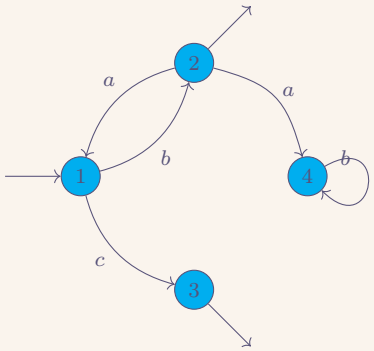


Figure 11 – Automate reconnaissant le langage $(ab)^* \cdot c \mid a(ba)^*$

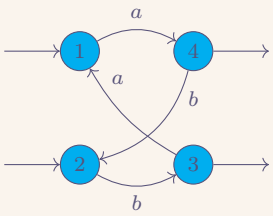


Figure 12 – Automate reconnaissant le langage $(a \mid baa)(bbaa)^* \mid (b \mid abb)(aabb)^*$

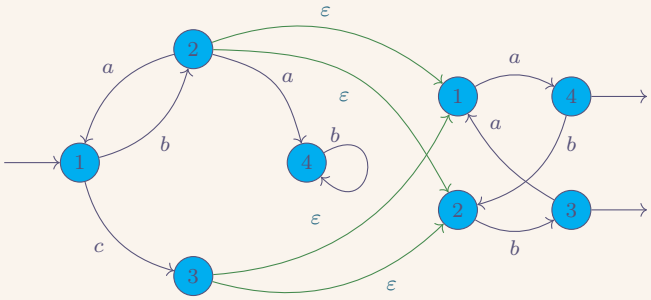


Figure 13 – Automate reconnaissant la concaténation des deux précédents

5.1 Cloture par concaténation

Propriété: Soient $\mathscr{A} = (\Sigma, Q, I, F, \delta)$ et $\mathscr{A}' = (\Sigma', Q', I', F', \delta')$ deux automates avec $Q \cap Q' = \varnothing$. Alors $\mathscr{L}(\mathscr{A}) \cdot \mathscr{L}(\mathscr{A}')$ est un langage reconnaissable. Il est d'ailleurs reconnu par l'automate $\mathscr{A}^{\cdot} = (\Sigma^{\cdot}, Q^{\cdot}, I^{\cdot}, F^{\cdot}, \delta^{\cdot})$ défini avec $\Sigma^{\cdot} = \Sigma \cup \Sigma'$, $Q^{\cdot} Q \cup Q'$, $I^{\cdot} = I$, $F^{\cdot} = F'$ et

$$\delta^{\cdot} = \{(q, \varepsilon, q) \mid q \in F,\, q' \in I'\}.$$

Preuve:
Montrons que $\mathscr{L}(\mathscr{A}^{\cdot}) = \mathscr{L}(\mathscr{A}) \cdot \mathscr{L}(\mathscr{A}')$. On procède par double inclusion.
"$\subseteq$" Soit $w \in \mathscr{L}(\mathscr{A}^{\cdot})$ et soit

$$Q \supseteq I = I^{\cdot} \ni q_0 \xrightarrow{u_1} q_1 \xrightarrow{u_2} q_2 \to \cdots \to q_{n-1} \xrightarrow{u_n} q_n \in F^{\cdot} = F' \subseteq Q$$

une exécution acceptante de $\mathscr{A}^{\cdot}$ telle que $\tilde{u} = w$. On pose $i_0 = \min\{i \in [\![1, n]\!] \mid q_i \in Q\}$. On a alors, $\forall i \in [\![1, i_0 - 1]\!]$, $q_i \in Q$. Montrons que $\forall i \in [\![i_0, n]\!]$, $q_i \in Q'$ par récurrence finie. On a $q_{i_0} \in Q'$. De plus, si $q_i \in Q'$ et $(q_i, u_{i+1}, q_{i+1}) \in \delta^{\cdot}$ donc $q_{i+1} \in Q'$. Inspectons $(Q \ni q_{i_0-1}, u_{i_0}, q_{i_0} \in Q') \in \delta^{\cdot}$. On sait que $(q_{i_0-1}, u_{i_0}, q_{i_0}) \notin \delta$ car $q_{i_0} \in Q'$; de même, $(q_{i_0-1}, u_{i_0}, q_{i_0}) \notin \delta'$ car $q_{i_0-1} \in Q$ donc $(q_{i_0-1}, u_{i_0}, q_{i_0}) \in \{(q, \varepsilon, q') \mid q \in Fnq' \in I'\}$. On a donc que $q_{i_0-1} \in F$ et $q_{i_0} \in I$. Ainsi

$$I \ni \underbrace{q_0 \xrightarrow{u_1} q_1 \to \cdots \to \overset{F}{\ni} q_i}_{} \xrightarrow{\varepsilon} \underbrace{\overset{I'}{\ni} q_{i_0} \xrightarrow{u_{i_0}} \to \cdots \to q_n}_{} \in F$$

est une exécution acceptante de $\mathscr{A}$ d'étiquette $u_1 \widetilde{\cdots} u_{i_0-1}$.

est une exécution acceptante de $\mathscr{A}'$ d'étiquette $u_{i_0+1} \widetilde{\cdots} u_n$.

donc $u_{|\widetilde{[\![1,i_0-1]\!]}} \in \mathscr{L}(\mathscr{A})$ et $u_{|\widetilde{[\![i_0+1,n]\!]}} \in \mathscr{L}(\mathscr{A}')$.

$$\begin{aligned} w = \tilde{u} &= u_{|[\![1,i_0-1]\!]} \cdot \widetilde{u_{i_0}} \cdot u_{|[\![i_0+1,n]\!]} \\ &= u_{|\widetilde{[\![1,i_0+1]\!]}} \cdot u_{|\widetilde{[\![i_0+1,n]\!]}} \end{aligned}$$

"$\supseteq$" Montrons que $\mathscr{L}(\mathscr{A}) \cdot \mathscr{L}(\mathscr{A}') \subseteq \mathscr{L}(\mathscr{A}^*)$. Soit $w \in \mathscr{L}(\mathscr{A}) \cdot \mathscr{L}(\mathscr{A}^{\cdot})$, il existe donc

$$I \ni q_0 \xrightarrow{u_1} q_1 \to \cdots \to q_n \in F$$

une exécution acceptante dans $\mathscr{A}$ d'étiquette $u_1 \widetilde{\cdots} u_n$. Il existe également

$$I' \ni q_{n+1} \xrightarrow{u_{n+2}} q_{n+1} \to \cdots \to q_m \in F$$

une exécution acceptante dans $\mathscr{A}'$ d'étiquette $u_{n+2} \widetilde{\cdots} u_m$. Or, $\delta \subseteq \delta^{\cdot}$ et $\delta' \subseteq \delta^{\cdot}$ donc $\forall i \in [\![1, n]\!]$, $(q_{i-1}, u_i, q_i) \in \delta^{\cdot}$ et $\forall i \in [\![n+2, m]\!]$, $(q_{i-1}, u_i, q_i) \in \delta^{\cdot}$. Or, $q_n \in F$ et $q_{n+1} \in I'$ donc $(q_n, \varepsilon, q_{n+1}) \in \delta^{\cdot}$. Finalement À faire : recopier.

□

5.2 Cloture par étoile

Propriété: Soit $\mathscr{A} = (\Sigma, Q, I, F, \delta)$ un automate fini. Alors $\mathscr{L}(\mathscr{A})^*$ est un automate reconnaissable, il est de plus reconnu par l'automate $\mathscr{A}_* = (\Sigma_*, Q_*, I_*, F_*, \delta_*)$ défini avec $\Sigma_* = \Sigma$, $Q_* = Q \cup \{V\}$ où $V \notin Q$. À faire : recopier ici...

Exemple:

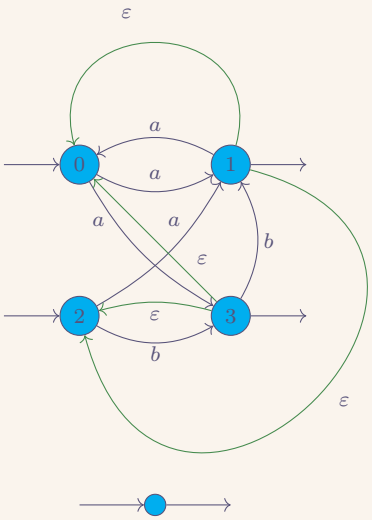


FIGURE 14 – Automate reconnaissant $\mathscr{L}(\mathscr{A})^*$

5.3 Cloture par union

Propriété: Soit $\mathscr{A} = (\Sigma, Q, I, F, \delta)$ et $\mathscr{A}' = (\Sigma', Q', I', F', \delta')$ deux automates avec $Q \cap Q' = \varnothing$. Alors, $\mathscr{L}5\mathscr{A}) \cup \mathscr{L}(\mathscr{A}')$ est un langage reconnaissable. Il est, de plus, reconnu par $\mathscr{A}^\cup = (\Sigma^\cup, Q^\cup, I^\cup, F^\cup, \delta^\cup)$ avec $\Sigma^\cup = \Sigma \cup \Sigma'$, $Q^\cup = Q \cup Q'$, $I^\cup = I \cup I'$, $F^\cup = F \cup F'$ et $\delta^\cup = \delta \cup \delta'$.

Preuve:
Montrons que $\mathscr{L}(\mathscr{A}^\cup) \subseteq \mathscr{L}(\mathscr{A}) \cup \mathscr{L}(\mathscr{A}')$. Supposons, sans perte de généralité que les automates $\mathscr{A}$ et $\mathscr{A}'$ sont sans ε-transitions. Soit $w \in \mathscr{L}(\mathscr{A}^\cup)$. Il existe une execution acceptante

$$I^\cup \ni q_0 \xrightarrow{w_1} q_1 \to \cdots \to q_{n-1} \xrightarrow{w_n} q_n \in F^\cup$$

avec $w = w_0 \dots w_n$.

Montrons que, en supposant $q_0 \in I$ sans perte de généralité, $\forall i \in [\![1, n]\!]$, $q_i \in Q$ de proche en proche.

On a donc $q_n \in Q \cap F^\cup = F$ et on a alors, pour tout $i \in [\![1, n]\!]$, $(q_{i-1}, w_i, q_i) \in \delta^\cup$. Or, $q_{i-1} \in Q$ et $(q_{i-1}, w_i, q_i) \in \delta'$ donc $(q_{i-1}, w_i, q_i) \in \delta$.

Finalement, $q_0 \xrightarrow{w_1} q_1 \to \cdots \to q_n$ est un exécution acceptante de $\mathscr{A}$ donc $w \in \mathscr{L}(\mathscr{A})$. □

REMARQUE:
Pour tout $a \in \Sigma$, $\{a\}$ est reconnaissable : par exemple,

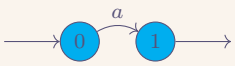


Figure 15 – Automate reconnaissant $\{a\}$ avec $a \in \Sigma$

Remarque:
$\varnothing$ est reconnaissable : par exemple,

Figure 16 – Automate reconnaissant $\varnothing$

Propriété: De ce qui précède, on en déduit que l'ensemble des langages reconnaissables par automates avec ε-transition est au moins l'ensemble des langages réguliers.

Théorème: Si $\mathscr{A}$ est un automate avec ε-transitions, il existe un automate $\mathscr{A}'$ un automate $\mathscr{A}'$ sans ε-transition tel que $\mathscr{L}(\mathscr{A}) = \mathscr{L}(\mathscr{A}')$.

Exemple:

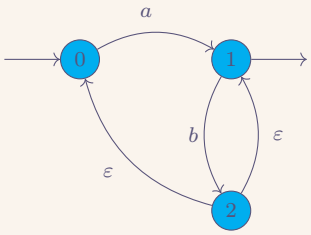


Figure 17 – Automate avec ε-transition

L'automate avec ε-transition ci-dessus peut être transformé en automate sans ε-transition comme celui ci-dessous.

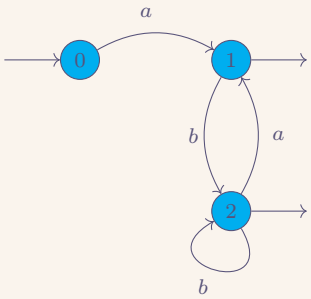


Figure 18 – Automate sans ε-transition

Propriété: Soit $\mathscr{A} = (\Sigma, Q, I, F, \delta)$ un automate avec ε-transitions. Soit $q_r \in Q$ un état de l'automate. Alors, l'automate $\mathscr{A}' = (\Sigma', Q', I', F', \delta')$ défini par $\Sigma' = \Sigma$,

$\mathcal{Q}' = \mathcal{Q}$, $I' = I$,

$$F' = F \cup \begin{cases} \{q \in \mathcal{Q} \mid (q, \varepsilon, q_r) \in \delta\} & \text{si } q_r \in F \\ \varnothing & \text{sinon,} \end{cases}$$

$$\begin{aligned} \delta' = (\delta \setminus \{(q, \varepsilon, q_r) \in \delta \mid q \in \mathcal{Q}\}) \\ \cup \{(q, a, q') \mid (q, \varepsilon, q_r) \in \delta \,\text{et}\, (q_r, a, q') \in \delta \,\text{et}\, a \in \Sigma\} \\ \cup \{(q, \varepsilon, q') \mid (q, \varepsilon, q_r) \in \delta \text{ et } (q_r, \varepsilon, q') \in \delta \text{ et } q_r \neq q' \in \mathcal{Q}\}, \end{aligned}$$

est tel que
- il n'y a pas d'ε-transitions entrant en q_r ;
- $\mathscr{L}(\mathscr{A}') = \mathscr{L}(\mathscr{A})$;
- si $q \in \mathcal{Q}$ n'a pas d'ε-transition entrante dans $\mathscr{A}$, il n'en a pas dans $\mathscr{A}'$.

Algorithme (de suppression des ε-transitions)**:**
À faire : Retaper algorithme proprement
Entrée : Un automate $\mathscr{A} = (\Sigma, \mathcal{Q}, I, F, \delta)$.
Sortie : Un automate équivalent à $\mathscr{A}$, sans ε-transitions.
Début
$\delta' \leftarrow \delta$
$F' \leftarrow F$
Tant qu'il existe $q \in \mathcal{Q}'$ avec ε-transition entrante dans δ' faire :
$(\Sigma', \mathcal{Q}', I', F', \delta') \leftarrow$ résultat de la proposition précédente avec $q_R = q$
Fin tant que
Retourne $(\Sigma', \mathcal{Q}', I', F', \delta')$
Fin

On a donc démontré que tout langage régulier peut être reconnu par un automate.

Exemple:
On veut, par exemple, reconnaître le langage $(a \cdot b)^* \cdot (a \mid b)$. À faire : Faire les automates

6 Théorème de Kleine

On s'intéresse à un autre ensemble de langages, les *langages locaux*.

6.1 Langages locaux

6.1.1 Définitions, propriétés

Définition (lettre préfixe, lettre suffixe, facteur de taille 2, non facteur)**:** Soit L un langage. On note l'ensemble $P(L)$ des lettres préfixes défini comme

$$\begin{aligned} P(L) &= \{\ell \in \Sigma \mid \exists w \in \ell, \exists v \in \Sigma^*, w = \ell \cdot v\} \\ &= \{\ell \in \Sigma,\ \{\ell\} \cdot \Sigma^* \cap L = \varnothing\}. \end{aligned}$$

On note l'ensemble $S(L)$ des lettres suffixes défini comme

$$S(L) = \{w_{|w|} \mid w \in L\} = \{\ell \in \Sigma \mid \Sigma^* \cdot \{\ell\} \cap L \neq \varnothing\}.$$

On note l'ensemble $F(L)$ des facteurs de taille 2 défini comme

$$F(L) = \{\ell_1 \cdot \ell_2 \in \Sigma^2 \mid \Sigma^* \cdot \{\ell_1, \ell_2\} \cdot \Sigma^* \cap L \neq \varnothing\}.$$

On note l'ensemble $N(L)$ des non-facteurs défini comme

$$N(L) = \Sigma^2 \setminus F(L).$$

On défini également l'ensemble

$$\Lambda(L) = L \cap \{\varepsilon\}.$$

Définition: Soit L un langage. On définit le langage local engendré par L comme étant

$$\rho(L) = \Lambda(L) \cup \Big(P(L) \cdot \Sigma^* \cap \Sigma^* \cdot S(L)\Big) \setminus \Sigma^* N(L) \Sigma^*.$$

Exemple:
Avec $L = \{aab, \varepsilon\}$, on a $P(L) = \{a\}$, $S(L) = \{b\}$, $F(L) = \{aa, ab\}$, $N(L) = \{ba, bb\}$, $\Lambda(L) = \{\varepsilon\}$. Et donc, on en déduit que

$$\rho(L) = \{\varepsilon\} \cup \{ab\} \cup \{aab\} \cup \cdots \{\varepsilon\} \cup \{a^n \cdot b \mid n \in \mathbb{N}^*\}.$$

Définition: Un langage est dit local s'il est son propre langage engendré i.e. $\rho(L) = L$.

Propriété: Soit L un langage. Alors, $\rho(L) \supseteq L$.

Preuve:
Soit $w \in L$. Montrons que $w \in \rho(L)$.
- Si $w = \varepsilon$, alors $\Lambda(L) = L \cap \{\varepsilon\} = \{\varepsilon\}$ donc $w \in \rho(L)$.
- Sinon, notons $w = w_1 w_2 \dots w_n$. On doit montrer que $w_1 \in P(L)$, $w_n \in S(L)$, et $\forall i \in [\![1, n-1]\!]$, $w_i w_{i+1} \in F(L)$. Par définition de ces ensembles, c'est vrai.

□

Propriété: Soit L de la forme

$$\lambda \cup (P\Sigma^* \cap \Sigma^* S) \setminus (\Sigma^* N \Sigma^*)$$

avec $\Lambda \subseteq \{\varepsilon\}$, $P \subseteq \Sigma$, $S \subseteq \Sigma$, et $N \subseteq \Sigma^2$. Alors $\rho(L) = L$.

Preuve:
On a $L \subseteq \rho(L)$. Montrons donc $\rho(L) \subseteq L$.

- Montrons que $\Lambda(L) \subseteq L$.
  - Si $\Lambda(L) = \varnothing$, alors ok.
  - Sinon, $\Lambda(L) = \{\varepsilon\} = L \cap \{\varepsilon\}$ donc $\varepsilon \in L$ et donc $\varepsilon \in \Lambda$ parce que ce n'est possible.
- Montrons que $P(L) \subseteq P$. Soit $\ell \in P(L)$. Soient $v \in \Sigma^*$, et $w \in L$ tels que $w = \ell v$. On a donc $w \notin \Lambda$, donc $w \in (P\Sigma^* \cap \Sigma^* S)$ et donc $\ell v = w \in P\Sigma^*$ et donc $\ell \in P$.
- De même, $S(L) \subseteq L$
- À faire à la maison : $N \subseteq N(L)$ (ou $F(L) \subseteq F$)

□

Corollaire: On a $\rho^2 = \rho$.

À faire : Figure ensembles langages locaux, réguliers, ...

Preuve:
$\rho(\varnothing) = \varnothing$ et $\rho(\Sigma^*) = \Sigma^*$. □

Remarque:
Un langage L est local si et seulement s'il existe $S \subseteq \Sigma$, $P \subseteq \Sigma$, $N \subseteq \Sigma^2$ tel que

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

Exemple:
Le langage $L = \{a\}$ est local avec $S = \{a\}$, $P = \{a\}$, $F = \varnothing$ et $\Lambda = \varnothing$.

Le langage $L = \{a, ab\}$ est local avec $S = \{b, a\}$, $P = \{a\}$, $F = \{ab\}$ et $\Lambda = \varnothing$.

Le langage $L = (ab)^*$ est local avec $S = \{b\}$, $P = \{a\}$, $F = \{ab, ba\}$. Soit $w \in \rho(L)$. Si $w = \varepsilon$, alors ok. Sinon, $w = abw_1$ et $w_1 \in \rho(L)$. Par récurrence, on montre que le langage est local.

Le langage $L = a \cdot (ab)^*$ n'est pas local.

6.1.2 Stabilité

Intersection

Propriété: Si L_1 et L_2 sont deux langages locaux, alors $L_1 \cap L_2$ est un langage local.

Preuve:
Soit $L_1 = \Lambda_1 \cup \big(P_1\Sigma^* \cap \Sigma^* S_1\big) \setminus \big(\Sigma^* N_1 \Sigma^*\big)$, et $L_2 = \Lambda_2 \cup \big(P_2\Sigma^* \cap \Sigma^* S_2\big) \setminus \big(\Sigma^* N_2 \Sigma^*\big)$. On pose $F_1 = \Sigma^2 \setminus N_1$ et $F_2 = \Sigma^2 \setminus N_2$. On pose alors $\Lambda_\cap = \Lambda_1 \cap \Lambda_2$; $P_\cap = P_1 \cap P_2$; $S_\cap = S_1 \cap S_2$; $F_\cap = F_1 \cap F_2$; $N_\cap = \Sigma^2 \setminus F_\cap$. On a

$$L_1 \cap L_2 = \Lambda_\cap \cup \big(P_\cap \Sigma^* \cap \Sigma^* S_\cap\big) \setminus \Sigma^* N_\cap \Sigma^*.$$

En effet,

$$\begin{aligned} L_1 \cap L_2 &= (\Lambda_1 \cap \Lambda_2) \\ &\quad \cap (\Lambda_1 \cap (P_2\Sigma^* \cap \Sigma^* S_2) \setminus \Sigma^* N_2 \Sigma^*) \\ &\quad \cap \big(((P_1\Sigma^* \cap \Sigma^* S_1) \setminus \Sigma^* N_1 \Sigma^*) \cap \Lambda_2\big) \\ &\quad \cap \big(((P_1\Sigma^* \cap \Sigma^* S_1) \setminus \Sigma^* N_1 \Sigma^*) \cap (P_2\Sigma^* \cap (P_2\Sigma^* \cap \Sigma^* S_2) \setminus \Sigma^* N_2 \Sigma^*\big) \\ &= (\Lambda_1 \cap \Lambda_2)\big((P_1 \cap P_2)\Sigma^* \cap \Sigma^*(S_1 \cap S_2)\big) \setminus \Sigma^*(N_1 \cap N_2)\Sigma^* \end{aligned}$$

□

Union

Contre-exemple:
Avec $L_1 = ab$ et $L_2 = ba$, on a $\Lambda_1 = \Lambda_2 = \varnothing$, $P_1 = \{a\}$, $P_2 = \{b\}$, $S_1 = \{b\}$, $S_2 = \{b\}$, $F_1 = \{ab\}$ et $F_2 = \{ba\}$. Le langage $L_1 \cup L_2 = \{ab, ba\}$ n'est pas local : en effet, on a $\Lambda = \varnothing$, $P = \{a, b\}$, $S = \{a, b\}$, et $F = ab, ba$. Le mot aba est donc dans le langage local engendré.

On doit donc ajouter une contrainte afin d'éviter ce type de contre-exemples. L'intersection des alphabets est vide.

Propriété: Soient L_1 un langage local sur un alphabet Σ_1 et L_2 un langage local sur un alphabet Σ_2 avec $\Sigma_1 \cap \Sigma_2 = \varnothing$. Alors $L_1 \cup L_2$ est local.

Preuve:
Soient Λ_1, S_1, P_1, N_1, F_1 tels que L_1 soit défini par $(\Lambda_1, S_1, P_1, N_1, F_1)$. De même, soient Λ_2, S_2, P_2, N_2, F_2 tels que L_2 soit défini par $(\Lambda_2, S_2, P_2, N_2, F_2)$. Construisons alors $\Lambda_\cup = \Lambda_1 \cup \Lambda_2$, $P_\cup = P_1 \cup P_2$, $S_\cup = S_1 \cup S_2$, $F_\cup = F_1 \cup F_2$ et $N_\cup = (\Sigma_1 \cup \Sigma_2)^2 \setminus F_\cup$. On note $\Sigma = \Sigma_1 \cup \Sigma_2$. Montrons alors que

$$L_1 \cup L_2 = \underbrace{\Lambda_\cup \cup (P_\cup \Sigma^* \cap \Sigma^* S_\cup) \setminus (\Sigma^* N_\cup \Sigma^*)}_{L_\cup}.$$

On procède par double-inclusion.
"$\subseteq$" Soit $w \in L_1 \cup L_2$.
Cas 1 $w = \varepsilon$, alors $\Lambda_1 = \{\varepsilon\}$ ou $\Lambda_2 = \{\varepsilon\}$ donc $L_1 \cup L_2 = \{\varepsilon\}$ et donc $w \in L_\cup$.
Cas 2 $w \neq \varepsilon$. On pose $w = w_1 \dots w_n$. Sans perte de généralité, on suppose $w \in L_1$ et $w \notin L_2$. D'où $w_1 \in P_1$ et $w_n \in S_1$. Et, pour $i \in [\![1, n-1]\!]$, $w_i w_{i+1} \in F_1$ donc $w_1 \in P_1 \cup P_2$, $w_n \in S_1 \cup S_2$ et $\forall i \in [\![1, n-1]\!]$, $w_i w_{i+1} \in F_1 \cup F_2$. D'où $w \in L_\cup$.
"$\supseteq$" Cas 1 $w = \varepsilon$ alors $w \in \Lambda_\cup = L_1 \cup L_2$ donc $w \in \Lambda_1$ ou $w \in \Lambda_2$ donc $w \in L_1$ ou $w \in L_2$.
Cas 2 $w \neq \varepsilon$. On pose $w = w_1 w_2 \dots w_n$ avec $w_1 \in P_\cup$, $w_n \in S_\cup$ et $\forall i \in [\![1, n-1]\!]$, $w_i w_{i+1} \in F_\cup$. Alors, sans perte de généralité, on suppose $w_1 \in \Sigma_1$. Montrons par récurrence que $\forall p \in [\![1, n]\!]$, $w_p \in \Sigma_1$.
— On sait que $w_1 \in \Sigma_1$ par hypothèse.
— On suppose que $w_p \in \Sigma_1$ avec $p < n$. Alors, $w_p w_{p+1} \in F_\cup = F_1 \cup F_2$. Or, $F_2 \subseteq (\Sigma_2)^2$ et $w_p \in \Sigma_1$ avec $\Sigma_1 \cap \Sigma_2 = \varnothing$ donc $w_{p+1} \in \Sigma_1$.
On conclut par récurrence que $\forall i \in [\![1, n]\!]$, $w_i \in \Sigma_i$. Or, $w_n \in S_1 \cup S_2$ et $S_2 \cap \Sigma_1 = \varnothing$ donc $w_n \in S_1$. De plus, pour $i \in [\![1, n-1]\!]$, $w_i w_{i+1} \in F_1 \cup F_2$ donc $w_i w_{i+1} \in F_1$ et donc $w \in L_1$.

□

Concaténation

Contre-exemple:
Avec $L_1 = \{ab\}$ et $L_2 = \{ab\}$, deux langages locaux, alors $L_1 \cdot L_2 = \{abba\}$ n'est pas local. En effet, $P = \{a\}$, $S = \{a\}$, $F = \{ab, bb, ba\}$; or $aba \notin L_1 \cdot L_2$.

Propriété: Soient L_1 un langage local sur un alphabet Σ_1 et L_2 un langage local sur un alphabet Σ_2, avec $\Sigma_1 \cap \Sigma_2 = \varnothing$. Alors $L_1 \cdot L_2$ est un langage local.

Preuve:
Soient Λ_1, S_1, P_1, N_1, F_1 définissant L_1 et soient Λ_2, S_2, P_2, N_2, F_2 définissant L_2. Construisons $\Lambda_\bullet = \Lambda_1 \cap \Lambda_2$, $P_\bullet = P_1 \cup \Lambda_1 \cdot P_2$, $S_\bullet = S_2 \cup \Lambda_2 \cdot S_1$, $F_\bullet = F_1 \cup F_2 \cup S_1 \cdot P_2$, $\Sigma = \Sigma_1 \cup \Sigma_2$. Montrons que

$$L_1 \cdot L_2 = \underbrace{\Lambda_\bullet \cup (P_\bullet \Sigma^* \cap \Sigma^* S_\bullet) \setminus \Sigma^* N_\bullet \Sigma^*}_{L^\bullet}.$$

On procède par double inclusion.
"$\subseteq$" Soit $w \in L_1 \cdot L_2$.
— Si $w = \varepsilon$, alors $\varepsilon \in L_1$ et $\varepsilon \in L_2$ donc $w = \varepsilon \in \Lambda_\bullet \subseteq L^\bullet$.
— Sinon, $w = u \cdot v$ avec $u \in L_1$ et $v \in L_2$. On sait que $|u| > 0$ ou $|v| > 0$.
— Si $u \neq \varepsilon$, alors $u = u_1 \dots u_p$ avec $p \geqslant 1$. On sait que $u_1 \in P_1$, $u_p \in S_1$ et, pour $i \in [\![1, p-1]\!]$, $u_1 u_{i+1} \in F_1$.
Sous-cas 1 Si $v = \varepsilon$, alors $\Lambda_2 = \{\varepsilon\}$, et donc $S_1 \subseteq S_\bullet$. Or, $P_1 \subseteq P_\bullet$ et

$F_1 \subseteq F_\bullet$. On en déduit que $w = u \in L^\bullet$.

Sous-cas 2 $v \neq \varepsilon$, alors $v = v_1 \dots v_q$ avec $v_1 \in P_2$, $v_q \in S_2$ et, pour $i \in [\![1, nq-1]\!]$, $v_i v_{i+1} \in F_2$. Or, $u_p v_1 \in S_1 \cdot P_2$ et donc $w = u \cdot v \in L^\bullet$.

— Si $u = \varepsilon$, on procède de la même manière.

"$\supseteq$" Soit $w \in L^\bullet$.

— Si $w = \varepsilon$, alors $\varepsilon \in \Lambda_\bullet$ et donc $\varepsilon \in L_1$ et $\varepsilon \in L_2$. D'où $\varepsilon \in L_1 \cdot L_2$.

— Sinon, on pose $w = w_1 \dots w_n$.

Sous-cas 1 Si $\{i \in [\![1,n]\!] \mid w_i \in \Sigma_2\} = \varnothing$, on a donc $\forall i \in [\![1,n]\!]$, $w_i \in \Sigma_1$. De plus, $w_1 \in P_\bullet$, $w_n \in S_\bullet$ et, pour $i \in [\![1, n-1]\!]$, $w_i w_{i+1} \in F_\bullet$. Or, $P_\bullet \cap \Sigma_1 = P_1$, $S_\bullet \cap \Sigma_1 = S_1$, $\Lambda_2 = \{\varepsilon\}$ et $\forall i \in [\![1, n-1]\!]$ $w_i w_{i+1} \in F_1$. On en déduit que $w = w_1 \dots w_n \cdot \varepsilon \in L_1 \cdot L_2$.

Sous-cas 2 Si $M = \{i \in [\![1,n]\!] \mid w_i \in \Sigma_2\} \neq \varnothing$. Soit $i_0 = \min(M)$. D'où

$$w = \underbrace{w_1 \dots w_{i_0-1}}_{\in \Sigma_1} \cdot \underbrace{w_{i_0}}_{\in \Sigma_2} \dots w_n.$$

Si, $\ell_1, \ell_2 \in F_\bullet$, et $\ell_1 \in \Sigma_2$, alors $\ell_2 \in \Sigma_2$. De proche en proche, on en déduit que $\forall i \in [\![i_0, n]\!]$, $w_i \in \Sigma_2$.

— Si $i_0 = 1$, alors $w_1 \in \Sigma_2$, $w_1 \in P_\bullet$, $w_1 \in P_2$, $w_1 \in P_2$, $\Lambda_1 = \{\varepsilon\}$, et $w_n \in S_\bullet \cap \Sigma_2$, et $w_n \in S_2$. De plus, pour $i \in [\![1, n-1]\!]$, $w_i w_{i+1} \in F_\bullet \cap (\Sigma_2)^2$ donc $w_i w_{i+1} \in F_2$. D'où $w = \varepsilon \cdot w_1 \dots w_n \in L_1 \cdot L_2$.

— Si $i_0 > 1$, alors $w_1 \in \Sigma_1 \cap P^\bullet = P_1$, $w_n \in \Sigma_2 \cap S^\bullet = S_2$, et $\forall i \in [\![1, i_0 - 2]\!]$, $w_i w_{i+1} \in F^\bullet \cap (\Sigma_1)^2 = F_1$. D'où $w_{i_0-1} w_{i_0} \in F^\bullet \cap \Sigma_1 \Sigma_2 = S_1 P_2$ donc $w_{i_0-1} \in S_1$ et $w_{i_0} \in P_2$ donc $w_1 \dots w_{i_0-1} \in L_1$. Finalement, $\forall i \in [\![i_0, n-1]\!]$, $w_i w_{i_1 \in F_\bullet \cap (\Sigma)^2}$ donc À faire : Finir la preuve.

□

Étoile

Propriété: Soit L un langage local, alors L^* est un langage local.

Preuve:
Soient Λ, P, S, F et N définissant L. Alors $\Lambda_* = \{\varepsilon\}$, $P_* = P$, $S_* = S$ et $F_* = F \cup S \cdot P$.
À faire : preuve à faire à la maison. □

Exemple:
Avec $L = \{a, b\}$, un langage local, on a $\Lambda_* = \{\varepsilon\}$, $P_* = S_* = \{a, b\}$ (car $P = \{a, b\} = S$), et $S_* = \{ab, ba, aa, bb\}$.

6.1.3 Expressions régulières linéaires

Définition: Un expression régulière est dite *linéaire* si chacune de ses lettres apparaît une fois au plus dans l'expression.

Exemple:

Oui	Non
a	aa
$a\|b$	$a\|ba$

Table 2 – Exemples et non-exemples d'expressions régulières linéaires

Exemple:
On définit une fonction booléenne permettant de vérifier si une expression régulière est li-

néaire :

$$\text{linéaire} : \left(\begin{array}{rcl} \varnothing & \mapsto & \top \\ \varepsilon & \mapsto & \top \\ a \in \Sigma & \mapsto & \top \\ e_1 \cdot e_2 & \mapsto & (\text{vars}(e_1) \cap \text{vars}(e_2) = \varnothing) \text{ et linéaire}(e_1) \text{ et linéaire}(e_2) \\ e_1 \mid e_2 & \mapsto & (\text{vars}(e_1) \cap \text{vars}(e_2) = \varnothing) \text{ et linéaire}(e_1) \text{ et linéaire}(e_2) \\ {e_1}^* & \mapsto & \text{linéaire}(e_1) \end{array} \right).$$

Propriété: Le langage d'une expression régulière est local.

Preuve:
Il nous suffit de montrer le résultat sur le cas de base.

$\mathscr{L}(\varnothing) : \varnothing$ qui correspond à $\Lambda = \varnothing$, $S = \varnothing$, $P = \varnothing$, $F = \varnothing$.

$\mathscr{L}(\varepsilon) = \{\varepsilon\}$ qui correspond à $\Lambda = \{\varepsilon\}$ et $S = P = F = \varnothing$.

$\mathscr{L}(a) = \{a\}$ qui correspond à $\Lambda = \varnothing$, $S = \{a\}$, $P = \{a\}$ et $F = \varnothing$. □

Remarque:
Les grandeurs Λ, P, S et F sont de plus définies individuellement par la table suivante.

e	Λ	P	S	F
$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
ε	$\{\varepsilon\}$	$\varnothing$	$\varnothing$	$\varnothing$
a	$\varnothing$	$\{a\}$	$\{a\}$	$\varnothing$
${e_1}^*$	$\{\varepsilon\}$	$P(e_1)$	$S(e_1)$	$F(e_1) \cup S(e_1) \cdot P(e_1)$
$e_1 \cdot e_2$	$\Lambda(e_1) \cap \Lambda(e_1)$	$P(e_1) \cup \Lambda(e_2) \cdot P(e_2)$	$S(e_2) \cup \Lambda(e_2) \cdot S(e_1)$	$F(e_1) \cup F(e_2) \cup S(e_1) \cdot P(e_2)$
$e_1 \mid e_2$	$\Lambda(e_1) \cup \Lambda(e_2)$	$P(e_1) \cup P(e_2)$	$S(e_1) \cup S(e_2)$	$F(e_1) \cup F(e_2)$

Table 3 – Construction de Λ, P, S et F dans différents cas