Chapitre 1

Langages réguliers et Automates

Hugo Salou MPI*

Dernière mise à jour le 31 octobre 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

```c
typedef struct {
  int carre;
  bool sign;
};
```

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) :

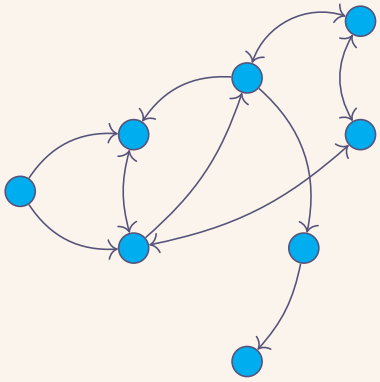


Figure 1 – États d'un ordinateur

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 langages, 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 $|w|$ pour un mot w. Si $|w| = n \in \mathbb{N}^*$, on indexe les lettres de w pour $(w_i)_{i\in[\![1,n]\!]}$ et on écrit alors

$$w = w_1 w_2 w_3 \dots w_n.$$

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

$$x = x_1 x_2 x_3 \dots x_n$$
$$y = y_1 y_2 y_3 \dots y_n.$$

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

$$x \cdot y = x_1 x_2 x_3 \dots x_n y_1 y_2 \dots y_n.$$

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 $\exists (v, w) \in \left(\Sigma^*\right)^2,\ y = v \cdot x \cdot w.$

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

$$a \not a\, a\, a \not a\, a \qquad \longrightarrow \qquad a\, a\, a\, a.$$

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

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

3 Langage régulier

3.1 Opérations sur les langages

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

Définition: Soient $L_1, L_2 \in \wp(\Sigma^*)$ 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 $\cdot$ (langages) a $\{\varepsilon\}$ pour neutre.
— L'opération $\cdot$ (langages) a $\varnothing$ pour élément absorbant :

$$\forall L \in \wp(\Sigma^*),\ L \cdot \varnothing = \varnothing \cdot L = \varnothing.$$

— 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* Kleene.

Remarque:
Avec L et K deux alphabets, on a

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

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

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 $L_1 \cup L_2 \in \mathrm{LR}$;

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

— 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 $L^* \in \mathrm{LR}$.

3.2 Expressions régulières

On représente $\varnothing$ l'ensemble vide, _|_ l'union de deux expressions régulières, _ · _ 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 $|(L(a), L(b)) =^? |(L(b), L(a))$? 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

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

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

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

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

■

Remarque (Notation):
Les notations $\mathrm{Reg}(\Sigma)$ et $\mathrm{Regexp}(\Sigma)$ 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.

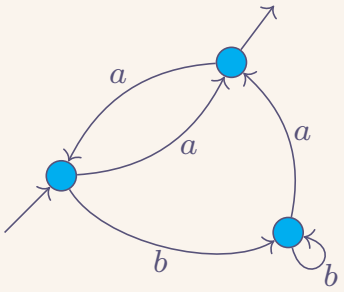


Figure 2 – Exemple d'automate

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.
- $\delta \subseteq Q \times \Sigma \times Q$.

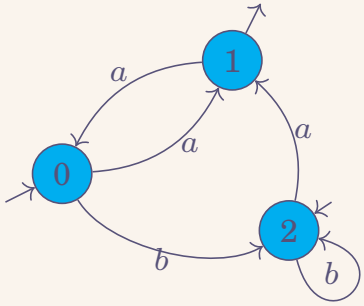


FIGURE 3 – Exemple d'automate (2)

Définition: On dit d'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* $\mathcal{A} = (Q, \Sigma, I, F, \delta)$ 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

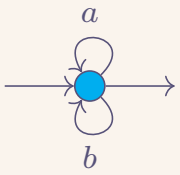
$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} q_3 \to \cdots \to q_n - 1 \xrightarrow{a_n} q_n.$$

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

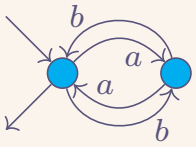
Lorsque $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \to \cdots \to q_{n-1} \xrightarrow{a_n} q_n$ est une suite de transitions d'un automate $\mathcal{A}$, on dit que le mot $a_1 a_2 \dots a_n$ est l'*étiquette* de cette transition.

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

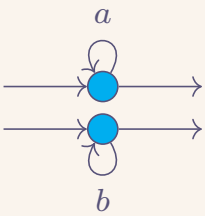
Définition: On dit d'un langage L qu'il est *reconnaissable* s'il existe un automate $\mathcal{A}$ tel que $\mathcal{L}(\mathcal{A}) = L$. On note $\mathrm{Rec}(\Sigma)$ l'ensemble des mots reconnaissables.



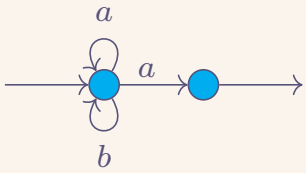
(a) Automate minimal pour $\mathscr{L}(\mathscr{A}) = \Sigma^*$



(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

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

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

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

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.

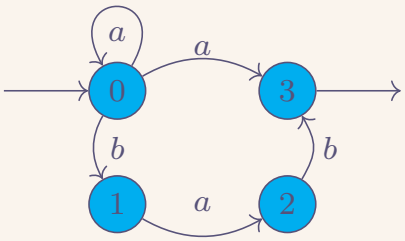


FIGURE 5 – Automate non déterministe ayant pour expression régulière $a^* \cdot (a \mid bab)$

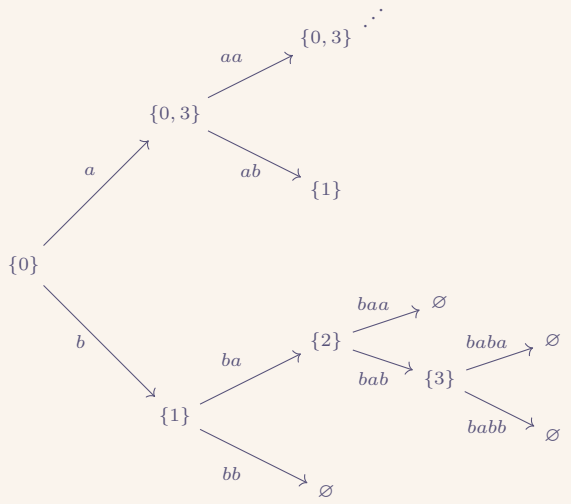


FIGURE 6 – Nœuds possibles par rapport à l'expression lue

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

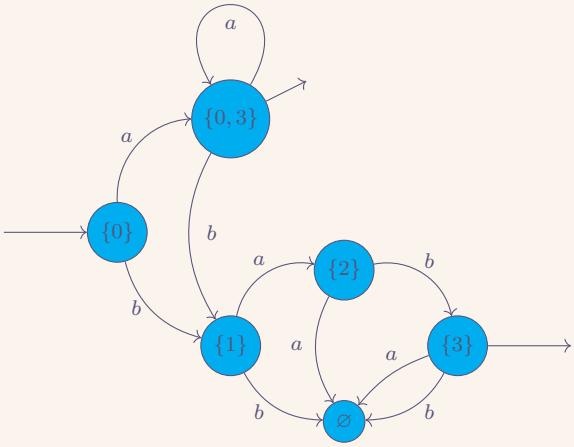


FIGURE 7 – Automate déterministe ayant pour expression régulière $a^* \cdot (a \mid bab)$

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

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

■

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

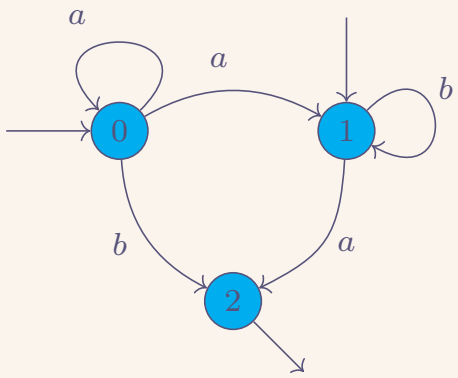


FIGURE 8 – Automate non déterministe

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 $\mathcal{A}'$ construit dans le théorème précédent est complet mais son nombre de nœud suit une exponentielle.

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

■

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

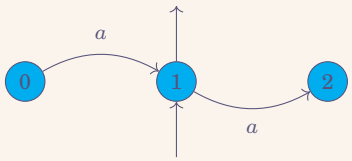


FIGURE 9 – Non-exemples d'états accessibles et co-accessibles

Définition: On dit d'un automate $\mathcal{A}$ 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 $\mathscr{L}(\mathscr{A}') = \mathscr{L}(\mathscr{A})$.

■

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.

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

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

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

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

■

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...

5.3 Cloture par union

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

■

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

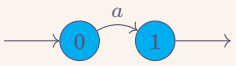


Figure 10 – Automate reconnaissant $\{a\}$ avec $a \in \Sigma$

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

Figure 11 – 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 $\mathcal{A}$ est un automate avec ε-transitions, alors il existe un automate $\mathcal{A}'$ sans ε-transition tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

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

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

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

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

Algorithme 1 Suppression des ε-transitions

Entrée un automate $\mathcal{A} = (\Sigma, \mathbb{Q}, I, F, \delta)$
Sortie un automate équivalent à $\mathcal{A}$ sans ε-transitions

$\delta' \leftarrow \delta$
$F' \leftarrow F$
$\mathbb{Q}' \leftarrow \mathbb{Q}$
tant que il existe $q \in \mathbb{Q}'$ avec une ε-transition entrante dans δ' **faire**
  $(\Sigma', \mathbb{Q}', I', F', \delta') \leftarrow$ résultat de la proposition précédente avec $q_R = q$
retourner $(\Sigma', \mathbb{Q}', I', F', \delta')$

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

6 Théorème de Kleene

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

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

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

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

■

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 $\rho(L) = L$.

■

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

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

■

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

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

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.

■

Union

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.

■

Concaténation

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.

■

Étoile

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

■

6.2 Expressions régulières linéaires

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

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

■

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

Remarque (Notation):
Si Σ_1 et Σ_2 sont deux alphabets et $\varphi : \Sigma_1 \to \Sigma_2$, alors on note $\tilde{\varphi}$ l'extension de φ aux mots de Σ_1^* :

$$\tilde{\varphi}(w_1 \dots w_n) = \varphi(w_1) \dots \varphi(w_n)$$

et, de plus, on note

$$\tilde{\varphi}(L) = \{\tilde{\varphi}(w) \mid w \in L\}.$$

Remarque:
On a $\tilde{\varphi}(L \cup M) = \tilde{\varphi}(L \cup M) = \tilde{\varphi}L \cup \tilde{\varphi}(M)$.

Propriété:

$$\tilde{\varphi}(L \cdot M) = \tilde{\varphi}(L) \cdot \tilde{\varphi}(M)$$

■

Définition: Soient $e \in \mathrm{Reg}(\Sigma_1)$, $\varphi : \Sigma_1 \to \Sigma_2$. On définit alors inductivement e_φ comme étant

$$\varnothing_\varphi = \varnothing \qquad a_\varphi = \varphi(a) \text{ si } a \in \Sigma_1 \qquad (e_1 \mid e_2)_\varphi = (e_1)_\varphi \mid (e_2)_\varphi$$
$$\varepsilon_\varphi = \varepsilon \qquad (e_1 \cdot e_2)_\varphi = (e_1)_\varphi \cdot (e_2)_\varphi \qquad (e_1^*)_\varphi = ((e_1)_\varphi)^*.$$

Propriété: Si $\varphi : \Sigma_1 \to \Sigma_2$ et $e \in \mathrm{Reg}(\Sigma_1)$, alors

$$\mathscr{L}(e_\varphi) = \tilde{\varphi}(\mathscr{L}(e)).$$

■

Propriété: Soit $e \in \mathrm{Reg}(\Sigma_1)$. Il existe $f \in \mathrm{Reg}(\Sigma)$ et $\varphi : \Sigma \to \Sigma_1$ tel que f est linéaire et $e = f_\varphi$.

■

6.3 Automates locaux

Définition (automate local, local standard)**:** Un automate $\mathscr{A} = (\Sigma, \mathbb{Q}, I, F, \delta)$ est dit local dès lors que pour out $\forall (q_1, q_2, \ell, q_3, q_4) \in \mathbb{Q} \times \mathbb{Q} \times \Sigma \times \mathbb{Q} \times \mathbb{Q}$,

$$(q_1, \ell, q_3) \in \delta \quad \text{et} \quad (q_2, \ell, q_4) \in \delta \qquad \Longrightarrow \qquad q_3 = q_4.$$

L'automate $\mathscr{A}$ est dit, de plus, standard lorsque $\mathrm{Card}(I) = 1$ et qu'il n'existe pas de transitions entrante en l'unique état initial q_0.

Propriété: Un langage est local si et seulement s'il est reconnu par un automate local standard.

■

Propriété: Soit $\mathscr{A} = (\Sigma, \mathbb{Q}, I, F, \delta)$ un automate et $\varphi : \Sigma \to \Sigma_1$. On pose

$$\delta' = \{(a, \varphi(\ell), q') \mid (q, \ell, q') \in \delta\}.$$

On pose $\mathscr{A}' = (\Sigma, \mathbb{Q}, I, F, \delta')$. On a $\mathscr{L}(\mathscr{A}') = \tilde{\varphi}(\mathscr{L}(\mathscr{A}))$.

Remarque (Notation):
On note A_φ, l'automate $(\varphi(E), \mathbb{Q}, I, F, \delta')$ où $\delta' = \{(q, \varphi(\ell), q') \mid (q, \ell, q') \in \delta\}$.

6.4 Algorithme de Berry-Sethi : les langages réguliers sont reconnaissables

Théorème: Tout langage régulier est reconnaissable. De plus, on a un algorithme qui calcule un automate le reconnaissant, à partir de sa représentation sous forme d'expression régulière.

Algorithme (Berry-Sethi)**:** Entrée : Une expression régulière e
Sortie : Un automate reconnaissant $\mathscr{L}(e)$

1. On linéarise e en f avec une fonction φ telle que $f_\varphi = e$.
2. On calcule inductivement $\Lambda(f)$, $S(f)$, $P(f)$, et $F(f)$.
3. On fabrique $\mathscr{A} = (\Sigma, \mathbb{Q}, I, F, \delta)$ un automate reconnaissant $\mathscr{L}(f)$.
4. On retourne $\mathscr{A}_\varphi$.

À faire : refaire la mise en page pour les algorithmes

6.5 Les langages reconnaissables sont réguliers

On fait le « sens inverse » : à partir d'un automate, comment en déduire le langage reconnu par cet automate ?

L'idée est de supprimer les états un à un. Premièrement, on rassemble les états initiaux en les reliant à un état i, et de même, on relie les états finaux à f. Pour une suite d'états, on concatène les lettres reconnus sur chaque transition :

Figure 12 – Succession d'états

De même, lors de « branches » en parallèles, on les concatène avec un $|$. En appliquant cet algorithme à l'automate précédent, on a

$$(aab) \cdot \Big((\varepsilon \mid aa^*) \mid (b \mid aa^*b) \cdot (b \mid aa^*b)^*(aa^* \mid \varepsilon) \Big).$$

Définition: Un automate généralisé est un quintuplet $(\Sigma, \mathbb{Q}, I, F, \delta)$ où
— Σ est un alphabet ;
— $\mathbb{Q}$ est un ensemble fini ;
— $I \subseteq \mathbb{Q}$;
— $F \subseteq \mathbb{Q}$;
— $\delta \subseteq \mathbb{Q} \times \mathrm{Reg}(\Sigma) \times \mathbb{Q}$, avec

$$\forall r \in \mathrm{Reg}(\Sigma),\ \forall (q, q') \in \mathbb{Q}^2,\ \mathrm{Card}(\{(q, r, q') \in \delta\}) \leqslant 1.$$

Définition (Langage reconnu par un automate généralisé)**:** Soit $(\Sigma, \mathbb{Q}, I, F, \delta)$ un automate généralisé. On dit qu'un mot w est reconnu par l'automate s'il existe une suite

$$q_0 \xrightarrow{r_1} q_1 \xrightarrow{r_2} q_2 \to \cdots \to q_{n-1} \xrightarrow{r_n} q_n$$

et $(u_i)_{i \in [\![1,n]\!]}$ tels que $\forall i \in [\![1, n]\!]$, $u_i \in \mathscr{L}(r_i)$ et $w = u_1 \cdot u_2 \cdot \ldots \cdot u_n$.

Définition: Un automate généralisé $(\Sigma, \mathbb{Q}, I, F, \delta)$ est dit « bien détouré [2] » si $I = \{i\}$ et $F = \{f\}$, avec $i \neq f$, tels que i n'a pas de transitions entrantes et f n'a pas de transitions sortantes.

Lemme: Tout automate généralisé est équivalent à un automate généralisé « bien détouré. » En effet, soit $\mathcal{A} = (\Sigma, \mathbb{Q}, I, F, \delta)$ un automate généralisé. Soit $i \notin \mathbb{Q}$ et $f \notin \mathbb{Q}$. On pose $\Sigma' = \Sigma$, $I' = \{i\}$, $F' = \{f\}$, $\mathbb{Q}' = \mathbb{Q} \cup \{i, f\}$ et

$$\delta' = \delta \cup \{(i, \varepsilon, q) \mid q \in I\} \cup \{(q, \varepsilon, f) \mid q \in F\}.$$

Alors, l'automate $\mathcal{A}' = (\Sigma', \mathbb{Q}', I', F', \delta')$ est équivalent à $\mathcal{A}$ et « bien détouré. »

Lemme: Soit $\mathcal{A} = (\Sigma, \mathbb{Q}, I, F, \delta)$ un automate généralisé « bien détouré » tel que $|\mathbb{Q}| \geqslant 3$. Alors il existe un automate généralisé « bien détouré » $\mathcal{A}' = (\Sigma, \mathbb{Q}', I, F, \delta')$ avec $\mathbb{Q}' \subsetneq \mathbb{Q}$ et $\mathscr{L}(\mathcal{A}) = \mathscr{L}(\mathcal{A}')$.

■

Théorème: Un langage reconnaissable est régulier.

■

Théorème (Kleene)**:** Un langage est régulier si et seulement s'il est reconnaissable. Et, on a donné un algorithme effectuant ce calcul dans les deux sens.

7 La classe des langages réguliers

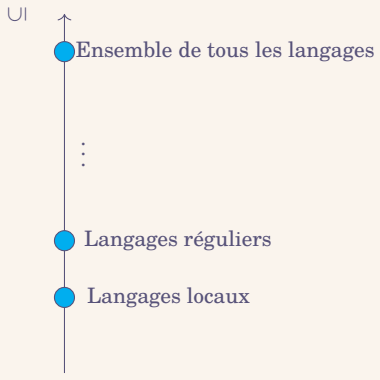


Figure 13 – Ensembles de langages

2. Cette notation n'est pas officielle.

Propriété: La classe des langages réguliers/reconnaissables est stable par passage au complémentaire.

■

Corollaire: On a la stabilité par intersection. En effet,

$$L \cap L' = \left(L^{\mathrm{c}} \cup (L')^{\mathrm{c}}\right)^{\mathrm{c}}$$

où L^{c} est le complémentaire de L.

Corollaire: Si L et L' sont deux langages réguliers (quelconques), alors $L \setminus L'$ est un langage régulier. En effet,

$$L \setminus L' = L \cap (L')^{\mathrm{c}}.$$

Corollaire: Si L et L' sont deux langages réguliers. Alors $L \triangle L'$ [3] est un langage régulier. En effet

$$L \triangle L' \overset{(\mathrm{def})}{=} (L \cup L') \setminus (L \cap L').$$

7.1 Limite de la classe/Lemme de l'étoile

Théorème (Lemme de l'étoile)**:** Soit L un langage reconnu par un automate à n états. Pour tout mot $u \in L$ de longueur supérieure ou égale à n, il existe trois mots x, y et z tels que

$$u = x \cdot y \cdot z, \qquad |x \cdot y| \leqslant n, \qquad y \neq \varepsilon, \qquad \text{et} \qquad \forall p \in \mathbb{N},\ x \cdot y^p \cdot z \in L.$$

■

Corollaire: Il y a des langages non réguliers/reconnaissables.

■

3. $\triangle$ est la différence symétrique

Annexe A. Comment prouver la correction d'un programme ?

Avec $\Sigma = \{a, b\}$. Comment montrer qu'un mot a au moins un a et un nombre pair de b.

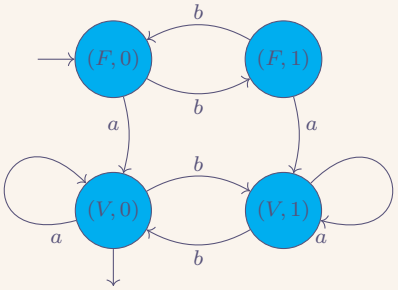


FIGURE 14 – Automate reconnaissant les mots valides

On veut montrer que

$$P_w : \text{« } \forall w \in \Sigma^*,\ \forall q \in \mathbb{Q},\ \text{(il existe une exécution par } w \text{ menant à } q) \iff w \text{ satisfait } I_q \text{ »}$$

où

$$I_{\underset{\mathbb{B}}{(v,} \ \underset{\{0,1\}}{r)}} \quad : \quad (|w|_a \geqslant 1 \iff v) \text{ et } (r = |w|_b \bmod 2).$$

On le montre par récurrence sur la longueur de w :

- " $\Longrightarrow$ " — Pour $w = \varepsilon$, alors montrons que $\forall q \in \mathbb{Q}$, il existe un exécution menant à q étiquetée par w (noté $\xrightarrow[\mathcal{A}]{w} q$) si et seulement si w satisfait I_q.
  - — $\xrightarrow[\mathcal{A}]{\varepsilon} (\boldsymbol{F}, 0)$ est vrai, de plus ε satisfait $I_{(\boldsymbol{F},0)}$;
  - — sinon si $q \neq (\boldsymbol{F}, 0)$, alors $\xrightarrow[\mathcal{A}]{\varepsilon} q$ est fausse, de plus ε ne satisfait pas I_q.
- — Supposons maintenant P_w vrai pour tout mot w de taille n. Soit $w = w_1 \dots w_n w_{n+1}$ un mot de taille $n+1$. Notons $\underline{w} = w_1 \dots w_n$. Montrons que P_w est vrai. Soit $q \in \mathbb{Q}$. Supposons $\xrightarrow[\mathcal{A}]{w} q$.
  - — Si $q = (\boldsymbol{F}, 0)$ et $w_{n+1} = b$. On a donc $\xrightarrow[\mathcal{A}]{w} (\boldsymbol{F}, 1)$, et, par hypothèse de récurrence, $\underline{w}$ satisfait. Donc $|\underline{w}|_a = 0$ et $|\underline{w}|_b \equiv 1\ [2]$ donc $|w|_a = 0$ et $|w|_b \equiv 0\ [2]$ donc w satisfait $I_{(\boldsymbol{F},0)}$.
  - — De même pour les autres cas.
- " $\Longleftarrow$ " Réciproquement, supposons que w satisfait I_q.
  - — Si $w = (\boldsymbol{V}, 0)$ et $w_{n+1} = a$. Alors,
    - — si $|\underline{w}|_a = 0$, alors $\underline{w}$ satisfait $I_{(\boldsymbol{F},0)}$. Par hypothèse de récurrence, on a donc $\xrightarrow[\mathcal{A}]{w} (\boldsymbol{F}, 0)$ et donc $\xrightarrow[\mathcal{A}]{w} (\boldsymbol{V}, 0)$.
    - — si $|\underline{w}|_b \geqslant 1$, alors $\underline{w}$ satisfait $I_{(\boldsymbol{V},0)}$ donc $\xrightarrow[\mathcal{A}]{w} (\boldsymbol{V}, 0)$ et donc $\xrightarrow[\mathcal{A}]{w} (\boldsymbol{V}, 0)$.
  - — De même pour les autres cas.

On a donc bien

$$\forall w \in \Sigma^*, \forall q \in \mathbb{Q},\ \xrightarrow[\mathcal{A}]{w} q \iff w \text{ satisfait } I_q.$$

Finalement,

$$\begin{aligned}\mathscr{L}(\mathscr{A}) &= \{w \in \Sigma^* \mid \exists f \in F,\ \xrightarrow[\mathscr{A}]{w} f\} \\ &= \{w \in \Sigma^* \mid \xrightarrow[\mathscr{A}]{w} (\boldsymbol{V}, 0)\} \\ &= \{w \in \Sigma^* \mid w \text{ satisfait } I_{(\boldsymbol{V},0)}\} \\ &= \{w \in \Sigma^* \mid |w|_a \geqslant 1 \text{ et } |w|_b \equiv 0 \ [2]\}\end{aligned}$$

Annexe B. Hors-programme

Définition: On appelle monoïde un ensemble M muni d'une loi "$\cdot$" interne associative admettant un élément neutre 1_M.

Définition: Étant donné deux monoïdes M et N, on appelle morphisme de monoïdes une fonction $\mu : M \to N$ telle que

1. $\mu(1_M) = 1_N$;
2. $\mu(x \cdot_M y) = \mu(x) \cdot_N \mu(y)$.

Définition: Un langage L est dit reconnu par un monoïde M, un morphisme $\mu : \Sigma^* \to M$ et un ensemble $P \subseteq M$ si $L = \mu^{-1}(P)$.

Théorème: Un langage est régulier si et seulement s'il est reconnu par un monoïde fini.

■