Théorie des langages et automates

Expressions régulières

Clément Moreau[1,2]

[1]BRED – Banque Populaire — clement.moreau@bred.fr

[2]Université de Tours

Sommaire

Table of Contents

Langage

- On rappelle qu'un langage L sur un alphabet Σ est un ensemble de mots $x \in \Sigma^*$.
- Un langage est dit *régulier* si il est reconnu par un automate fini.
- Une expression régulière E est une description algébrique d'un langage. On note celui-ci $L(E)$.

Algèbre de Kleene

Définition

Une algèbre de Kleene est une structure algébrique : $\mathcal{K} = (\Sigma, +, \cdot, *, \emptyset, \varepsilon)$ dotée des axiomes suivants :

A1	$a + \emptyset = a$	A7	$a(b + c) = ab + ac$
A2	$a + b = b + a$	A8	$(a + b)c = ac + bc$
A3	$(a + b) + c = a + (b + c)$	A9	$a(bc) = (ab)c$
A4	$a + a = a$	A10	$(a + b)^* = (a^* b)^* a^*$
A5	$a\emptyset = \emptyset a = \emptyset$	A11	$(ab)^* = \varepsilon + a(ba)^* b$
A6	$a\varepsilon = \varepsilon a = a$	A12	$(a^*)^* = a^*$

où $a, b, c \in \Sigma$.

Algèbre de Kleene

Expression régulière

On appelle *expression régulière* (ER) toute expression E sur $\mathcal{K}$ dont la formule s'écrit au moyen d'opérateurs de l'algèbre de Kleene et obéissant aux règles inductives suivantes :

- Tout symbole $a \in \Sigma$ est une expression régulière,
- Si $E \in \mathcal{K}$, alors $(E)^* \in \mathcal{K}$
- Si $E, F \in \mathcal{K}$, alors $(E + F) \in \mathcal{K}$
- Si $E, F \in \mathcal{K}$, alors $(EF) \in \mathcal{K}$

Exemples

- $E_1 = (0+1)^*01(0+1)^* \Leftrightarrow L(E_1) = \{x01y | x, y \in \{0,1\}*\}$
- $E_2 = ('-' + \varepsilon)(\llbracket 1,9 \rrbracket \llbracket 0,9 \rrbracket^* + 0)(\varepsilon + ('.' \llbracket 0,9 \rrbracket^* \llbracket 1,9 \rrbracket)) \Leftrightarrow L(E_2) = \mathbb{D}$

Algèbre de Kleene

Deux expressions régulières E, F sont équivalentes si et seulement si :

$$L(E) \subseteq L(F) \wedge L(F) \subseteq L(E)$$

On raisonne par double inclusion, ou par réduction à l'aide des axiomes vus précédemment.

Exemples

Montrer que les expressions régulières suivantes sont équivalentes :

1. $a^* = (aa)^* + a(aa)^*$
2. $a(ba)^* = (ab)^* a$

Équivalence entre automate et ER

Théorème

Pour toute expression régulière $E \in \mathcal{K}$, $L(E) \in \mathbf{Reg}$.

Corollaire

- Toute expression régulière peut-être convertie en automate fini.
  - Par l'algorithme de Thomson
  - Par dérivée partielle d'Antimirov
- Tout automate fini peut-être converti en ER.
  - Par méthode d'élimination d'états de Brzozowski et McCluskey
  - Par les équations d'Arden

Table of Contents

Équation d'Arden

Lemme d'Arden

Soient A, B deux langages sur un alphabet Σ. Soit l'équation :

$$X = AX + B$$

Le langage $L = A^*B$ est le plus petit langage qui est solution.

Équation d'Arden

Lemme d'Arden

Soient A, B deux langages sur un alphabet Σ. Soit l'équation :

$$X = AX + B$$

Le langage $L = A^*B$ est le plus petit langage qui est solution.

Démonstration.

1. L est solution. En effet, on a :
   $$\begin{aligned} AL + B &= A(A^*B) + B \\ &= (AA^* + \varepsilon)B \\ &= A^*B = L. \end{aligned}$$
2. Le langage A^*B est la plus petite solution.
   En développant l'équation et par récurrence, on obtient $X = A^{N+1}X + \sum_{n=0}^{N} A^nB$.
   Dès lors, $X = \lim_{N\to\infty} A^{N+1} + \sum_{n=0}^{N} A^nB = A^*B$.

□

Conversion AFN $\rightarrow$ ER

Soit $N = (Q, \Sigma, \Delta, S, F)$ un automate fini.
Pour $q \in Q, X_q$ est le langage des mots acceptés avec q comme état de départ.

$$X_q = \begin{cases} \sum\limits_{(q,a,p)\in\delta} aX_p + \varepsilon & \text{si } q \in F \\ \sum\limits_{(q,a,p)\in\delta} aX_p & \text{sinon} \end{cases}$$

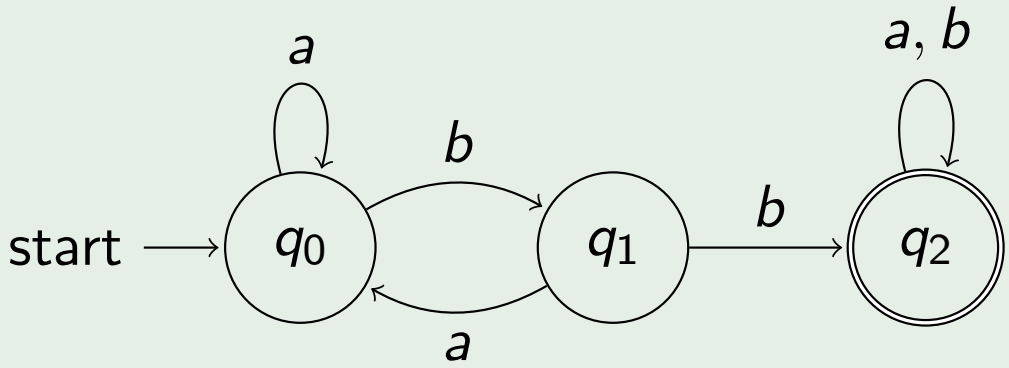
Où $(q, a, p) \in \delta \Leftrightarrow \delta(q, a) = p$.
On a alors :

$$L(N) = \sum_{s\in S} X_s$$

Conversion AFN $\rightarrow$ ER

Exemples

Soit l'automate N suivant :



On cherche à résoudre le système d'équations d'Arden $(\mathcal{S})$ suivant :

$$(\mathcal{S}) \Leftrightarrow \begin{cases} L_0 = aL_0 + bL_1 & (1) \\ L_1 = aL_0 + bL_2 & (2) \\ L_2 = aL_2 + bL_2 + \varepsilon & (3) \end{cases}$$

Conversion AFN $\rightarrow$ ER

Exemples

D'après le lemme d'Arden, l'équation (1) $\Leftrightarrow L_0 = a^* L_1$. Il vient que :

$$(\mathcal{S}) \Leftrightarrow \begin{cases} L_0 = a^* b L_1 & (1) \\ L_1 = a^+ b L_1 + b L_2 & (2) \\ L_2 = (a+b)^* & (3) \end{cases}$$

Conversion AFN $\rightarrow$ ER

Exemples

D'après le lemme d'Arden, l'équation (1) $\Leftrightarrow L_0 = a^* L_1$. Il vient que :

$$(\mathcal{S}) \Leftrightarrow \begin{cases} L_0 = a^* b L_1 & (1) \\ L_1 = a^+ b L_1 + b L_2 & (2) \\ L_2 = (a+b)^* & (3) \end{cases}$$

$$\Leftrightarrow \begin{cases} L_0 = a^* b L_1 & (1) \\ L_1 = (a^+ b)^* b L_2 & (2) \\ L_2 = (a+b)^* & (3) \end{cases}$$

Conversion AFN $\rightarrow$ ER

Exemples

D'après le lemme d'Arden, l'équation (1) $\Leftrightarrow L_0 = a^* L_1$. Il vient que :

$$
\begin{aligned}
(\mathcal{S}) &\Leftrightarrow \begin{cases} L_0 = a^* b L_1 & (1) \\ L_1 = a^+ b L_1 + b L_2 & (2) \\ L_2 = (a+b)^* & (3) \end{cases} \\
&\Leftrightarrow \begin{cases} L_0 = a^* b L_1 & (1) \\ L_1 = (a^+ b)^* b L_2 & (2) \\ L_2 = (a+b)^* & (3) \end{cases} \\
&\Leftrightarrow \begin{cases} L_0 = a^* b (a^+ b)^* b (a+b)^* & (1) \\ L_1 = (a^+ b)^* b (a+b)^* & (2) \\ L_2 = (a+b)^* & (3) \end{cases}
\end{aligned}
$$

On a finalement

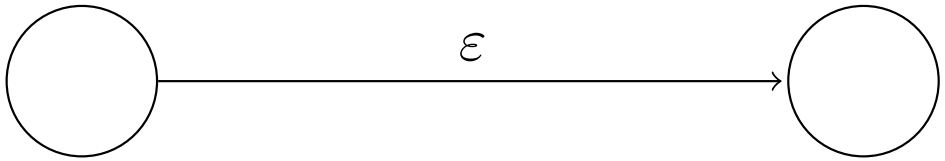
$$L(N) = a^* b (a^+ b)^* b (a+b)^*$$

Table of Contents

Algorithme de Thomson

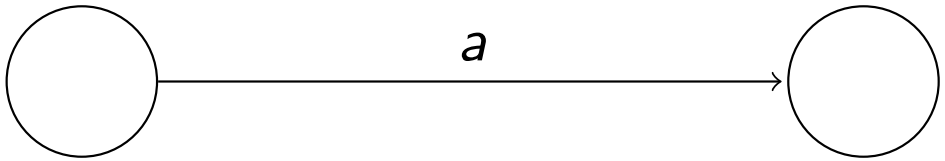
L'algorithme de Thomson construit un automate équivalent à une expression régulière E par induction :

- Si $E = \varepsilon$



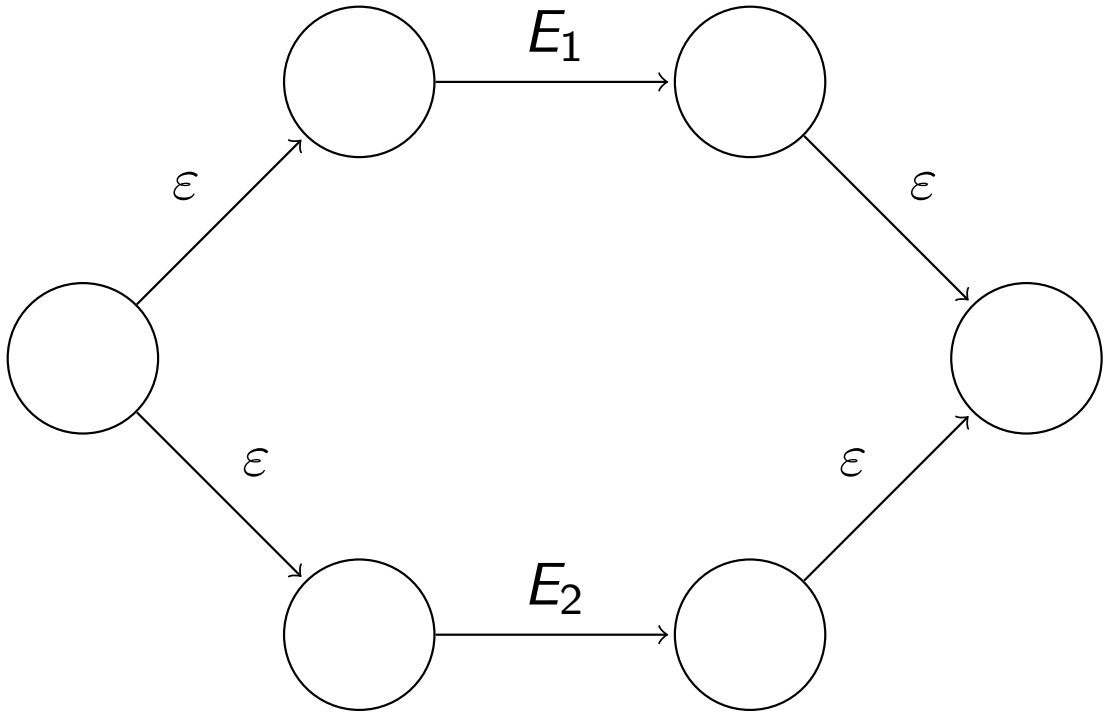
- Si $E = \emptyset$

- Si $E = a \in \Sigma$

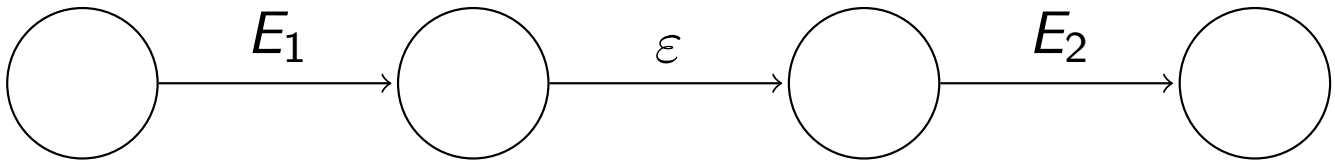


Algorithme de Thomson

- Si $E = E_1 + E_2$

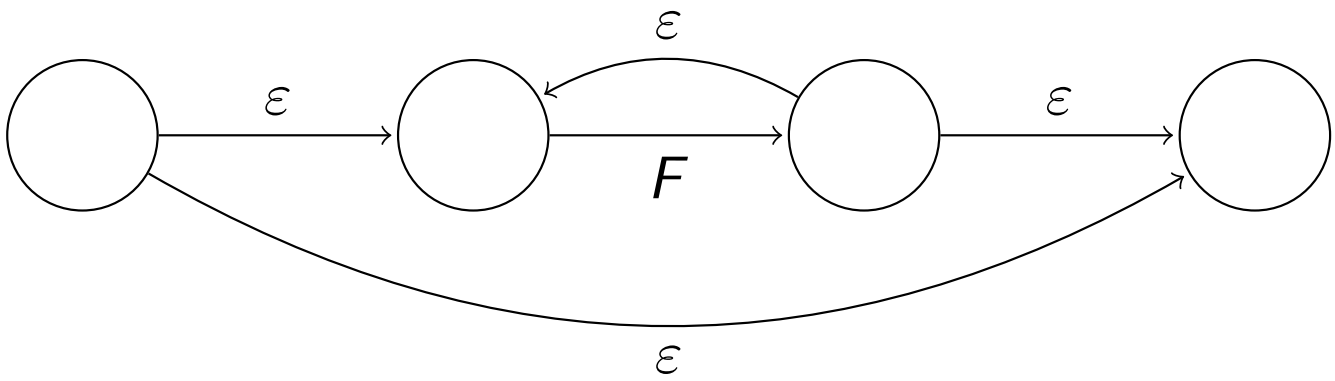


- Si $E = E_1 E_2$



Algorithme de Thomson

- Si $E = (F^*)$



Exemples

1. Appliquer l'algorithme de Thomson sur l'expression régulière :

$$E = (0 + 1)^* 1 (0 + 1)$$

2. Supprimer les ε-transitions inutiles.

Table of Contents

Pattern matching et expressions régulières

- Les expressions régulières ont un intérêt pratique pour le contrôle de format de documents et la *recherche de motif* dans un texte.
- Elles sont couramment utilisées au sein des différents langages de programmation (Java, C, Python, etc.) et des systèmes de type UNIX.

Regex syntaxe

Symbole	Description
.	N'importe quel symbole
[...]	Ensemble de caractères (Ex : [0-9a-z], [0-Z])
[^...]	Ensemble complémentaire
^	Début du mot
$	Fin de mot
|	Ou
(...)	Groupe de capture
*	0, 1 ou plusieurs occurrences
+	1 ou plusieurs occurrences
?	0 ou 1 occurrence
{...}	Comptage (Ex. $a\{3\} = aaa$, $a\{1,5\} = \{a, aa, ..., aaaaa\}$)

Attention

Cette symbologie peut varier selon les langages. Elle est valable ici pour Java.

Regex syntaxe

Symbole	Description
\d	Ensemble des chiffres
\s	Ensemble des espaces
\t	Tabulation
\w	Ensemble alphanumérique (chiffres, lettres et _)
\n	Saut de ligne
\p{L}	Ensemble des lettres
[[:alpha :]]	Ensemble des lettres
[[:digit :]]	Ensembles des chiffres
[[:alnum :]]	Ensemble des lettres et chiffres et _
[[:space :]]	Ensemble des espaces
[[:punct :]]	Ensembles des symboles de ponctuation
[[:lower :]]	Ensemble des lettres minuscules
[[:upper :]]	Ensemble des lettres majuscules

Regex

Exemples

Donner le sens des regex ou l'expression correspondante :

- \d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}

Regex

Exemples

Donner le sens des regex ou l'expression correspondante :

- \d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}
- <([a-z]+)[^>]>

Regex

Exemples

Donner le sens des regex ou l'expression correspondante :

- \d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}
- <([a-z]+)[^>]>
- Un numéro de téléphone français

Regex

Exemples

Donner le sens des regex ou l'expression correspondante :

- \d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}
- <([a-z]+)[^>]>
- Un numéro de téléphone français
- Une adresse mail de l'université de Tours.

Pattern matching et expressions régulières

Il existe plusieurs méthodes en Java selon l'utilité qu'on a des regex :

- Vérifier le bon format d'un texte.
- Chercher un fragment obéissant à un pattern donné.
- Remplacer ou extraire de l'information.

```
import java.util.regex.Pattern;

public class Regex {
   public static void main(String[] args) {
      String s = ...; // Some string
      String regex = ...; // Some regex
      Pattern.matches(regex, s); // s match the regex
      Pattern.compile(regex).matcher(s).find() // s contains a substring
          corresponding regex
      s.replaceAll(regex, _s) // remplace all substrings maching regex
          by _s in s
   }
}
```

Pattern matching et expressions régulières

Sous le pont Mirabeau coule la Seine
Et nos amours
Faut-il qu'il m'en souvienne
La joie venait toujours après la peine
[...]

L'amour s'en va comme cette eau courante
L'Amour s'en va
Comme la vie est lente
Et comme l'Espérance est violente

Exemples

Déterminer les fragments de code permettant de :

1. Trouver les mots qui comm par une majuscule mais qui ne sont pas des débuts de phrase.

Pattern matching et expressions régulières

Sous le pont Mirabeau coule la Seine
Et nos amours
Faut-il qu'il m'en souvienne
La joie venait toujours après la peine
[...]

L'amour s'en va comme cette eau courante
L'Amour s'en va
Comme la vie est lente
Et comme l'Espérance est violente

Exemples

Déterminer les fragments de code permettant de :

1. Trouver les mots qui comm par une majuscule mais qui ne sont pas des débuts de phrase.
2. Remplacer le mot "lente" par "rapide".

Pattern matching et expressions régulières

Sous le pont Mirabeau coule la Seine
Et nos amours
Faut-il qu'il m'en souvienne
La joie venait toujours après la peine
[...]

L'amour s'en va comme cette eau courante
L'Amour s'en va
Comme la vie est lente
Et comme l'Espérance est violente

Exemples

Déterminer les fragments de code permettant de :

1. Trouver les mots qui comm par une majuscule mais qui ne sont pas des débuts de phrase.
2. Remplacer le mot "lente" par "rapide".
3. Vérifier que le texte ne contient pas de ponctuation (hormis [...]).

Pattern matching et expressions régulières

Sous le pont Mirabeau coule la Seine
Et nos amours
Faut-il qu'il m'en souvienne
La joie venait toujours après la peine
[...]

L'amour s'en va comme cette eau courante
L'Amour s'en va
Comme la vie est lente
Et comme l'Espérance est violente

Exemples

Déterminer les fragments de code permettant de :

1. Trouver les mots qui comm par une majuscule mais qui ne sont pas des débuts de phrase.
2. Remplacer le mot "lente" par "rapide".
3. Vérifier que le texte ne contient pas de ponctuation (hormis [...]).
4. Vérifier que la première et la dernière phrase du premier couplet sont des rimes en "eine".

Pattern matching et expressions régulières

Sous le pont Mirabeau coule la Seine
Et nos amours
Faut-il qu'il m'en souvienne
La joie venait toujours après la peine
[...]

L'amour s'en va comme cette eau courante
L'Amour s'en va
Comme la vie est lente
Et comme l'Espérance est violente

Exemples

Déterminer les fragments de code permettant de :

1. Trouver les mots qui comm par une majuscule mais qui ne sont pas des débuts de phrase.
2. Remplacer le mot "lente" par "rapide".
3. Vérifier que le texte ne contient pas de ponctuation (hormis [...]).
4. Vérifier que la première et la dernière phrase du premier couplet sont des rimes en "eine".

Pattern matching et expressions régulières

```
// s equals the poem

/** Question 1 **/
Pattern p = Pattern.compile("[^\n[A-z]][A-Z][a-z]+");
Matcher m = p.matcher(s);
while (m.find()) {
    System.out.println(s.substring(m.start(0)+1, m.end(0)));
}

/** Question 2 **/
s.replaceAll(" lente", " rapide");

/** Question 3 **/
Pattern.compile("[?!.,;]").matcher(s).find() &&
    !Pattern.compile("\\[\\.\\.\\.\\]").matcher(s).find();

/** Question 4 **/
Pattern.matches("(\n|.)+eine\n(\n|.)+eine\n\\[\\.\\.\\.\\](\\n|.)*", s);
```

Table of Contents

Langage rationnel

Certains langages ne sont pas reconnaissables. On ne peut pas les exprimer à l'aide d'un automate.

Exemples

- $L_1 = \{a^n b^n | n \in \mathbb{N}\}$
- $L_2 = \{w \in \{a, b\}^* | w$ possède autant de a que de $b\}$
- $L_3 = \{w \in \{a, b\}^* | w$ possède un nombre différent de a et de $b\}$

Comment montrer qu'un langage n'est pas reconnaissable ?

Langage rationnel

Certains langages ne sont pas reconnaissables. On ne peut pas les exprimer à l'aide d'un automate.

Exemples

- $L_1 = \{a^n b^n | n \in \mathbb{N}\}$
- $L_2 = \{w \in \{a, b\}^* | w$ possède autant de a que de $b\}$
- $L_3 = \{w \in \{a, b\}^* | w$ possède un nombre différent de a et de $b\}$

Comment montrer qu'un langage n'est pas reconnaissable ?

Réponse : Par l'absurde, à l'aide du lemme de l'étoile ou grâce aux propriétés de clôture.

Langages régulier

Lemme de l'étoile

Soit $L \in \mathbf{Reg}\ (\Sigma^*)$. Soit $N \geq 0$ et $w \in L$, $\exists x, y, z$ tel que $w = xyz$ avec $0 < |y| \leq N$, alors $\forall n \geq 0, xy^n z \in L$

Précisons que L est supposé de cardinal infini.
Le lemme de l'étoile énonce que tout mot assez long et provenant de L comporte une partie non vide qui peut être itérée à l'infini.

Lemme de l'étoile

Démonstration.

Soit $L \in \mathbf{Reg}$, d'après le théorème de Kleene, il existe un automate fini M tel que $L(M) = L$. Soit N le nombre d'états de M.

Lemme de l'étoile

Démonstration.

Soit $L \in \mathbf{Reg}$, d'après le théorème de Kleene, il existe un automate fini M tel que $L(M) = L$. Soit N le nombre d'états de M.
Soit $w \in L$, tel que $|w| \geq N$. On sait également que $w \in L(M)$, c'est-à-dire qu'il existe un chemin $q_0 \xrightarrow{w} q_t$ avec q_0 comme état initial et q_t appartenant aux états finaux.

Lemme de l'étoile

Démonstration.

Soit $L \in \mathbf{Reg}$, d'après le théorème de Kleene, il existe un automate fini M tel que $L(M) = L$. Soit N le nombre d'états de M.
Soit $w \in L$, tel que $|w| \geq N$. On sait également que $w \in L(M)$, c'est-à-dire qu'il existe un chemin $q_0 \xrightarrow{w} q_t$ avec q_0 comme état initial et q_t appartenant aux états finaux.

On pose $w = a_1 a_2 ... a_N w'$ avec $a_{i \in [\![1,N]\!]} \in \Sigma$ et $w' \in \Sigma^*$. Soient $q_1, q_2, ..., q_N$ les états successifs atteints au cours de la lecture des N premiers symboles. On a donc :

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} ... \xrightarrow{a_N} q_N \xrightarrow{w'} q_t$$

Lemme de l'étoile

Démonstration.

Soit $L \in \mathbf{Reg}$, d'après le théorème de Kleene, il existe un automate fini M tel que $L(M) = L$. Soit N le nombre d'états de M.
Soit $w \in L$, tel que $|w| \geq N$. On sait également que $w \in L(M)$, c'est-à-dire qu'il existe un chemin $q_0 \xrightarrow{w} q_t$ avec q_0 comme état initial et q_t appartenant aux états finaux.

On pose $w = a_1 a_2 ... a_N w'$ avec $a_{i \in [\![1,N]\!]} \in \Sigma$ et $w' \in \Sigma^*$. Soient $q_1, q_2, ..., q_N$ les états successifs atteints au cours de la lecture des N premiers symboles. On a donc :

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} ... \xrightarrow{a_N} q_N \xrightarrow{w'} q_t$$

Comme M possède N états et comme de q_0 à q_N on parcours $N+1$ états, alors d'après le principe des tiroirs, il existe deux entiers i, j tels que $0 \leq i < j \leq N$ avec $q_i = q_j$. On pose q cet état et $x = a_1 ... a_i, y = a_{i+1} ... a_j, z = a_{j+1} ... a_N w'$. On a alors le chemin de w peut être factorisé tel que :

$$q_0 \xrightarrow{x} q \xrightarrow{y} q \xrightarrow{z} q_t$$

On a exhibé un cycle au niveau du mot y, il vient que $\forall n \geq 0, xy^n z \in L$. De plus, on a $|y| = j - i$, donc $0 < |y| \leq N$. □

Lemme de l'étoile

Exemples

On rappelle que $L_1 = \{a^n b^n | n \geq 0\}$. Soit :

$$w = \underbrace{aa...a}_{N}\underbrace{bb...b}_{N} = a_1...a_N b_1...b_N$$

On suppose que L_1 est régulier.
Posons :$x = a_1...a_{N-k}, y = a_{N-k+1}...a_N$ et $z = b_1...b_N$ tel que $w = xyz$ et $|y| = k$.
Alors, d'après le lemme de l'étoile : $\forall n \geq 0, xy^n z \in L$.
Il vient que $N = N + k(n-1) \Leftrightarrow 0 = k(n-1)$
Or $k > 0$, la propriété n'est pas valable pour tout $n \geq 0$. L'hypothèse de départ est absurde, L_1 n'est pas régulier.

Théorèmes de clôture

On montre que les autres langages ne sont pas reconnaissables grâce aux propriétés de clôture.

Théorème : Clôture par $\cap$ et $\neg$

- Soient L_1, L_2, L_3 tels que $L_1 \cap L_2 = L_3 \notin \textbf{Reg}$, alors $L_1 \notin$ ou $L_2 \notin \textbf{Reg}$.
- Soit $L \notin \textbf{Reg}$, alors $\neg L \notin \textbf{Reg}$.

Théorèmes de clôture

On montre que les autres langages ne sont pas reconnaissables grâce aux propriétés de clôture.

Théorème : Clôture par $\cap$ et $\neg$

- Soient L_1, L_2, L_3 tels que $L_1 \cap L_2 = L_3 \notin \textbf{Reg}$, alors $L_1 \notin$ ou $L_2 \notin \textbf{Reg}$.
- Soit $L \notin \textbf{Reg}$, alors $\neg L \notin \textbf{Reg}$.

Exemples

On sait que $L_1 \notin$ Reg.

- $L_2 \cap a^*b^* = L_1 \longrightarrow$ Donc L_2 n'est pas reconnaissable.
- $\neg L_3 = L_2 \longrightarrow$ Donc L_3 n'est pas reconnaissable.

Théorèmes de clôture

Morphisme

Soient deux monoïdes $(E, \star, e)$ et $(F, *, f)$. Un *morphisme* $\varphi : E \to F$ est une application qui vérifie :

- $\forall x, y \in E, \varphi(x \star y) = \varphi(x) * \varphi(y)$
- $\varphi(e) = f$

Exemples

Soit $\Sigma = \{0, 1\}$ et le morphisme $\varphi : \Sigma^* \to \Sigma^*$ tel que

$$\begin{cases} \varphi(0) = 01 \\ \varphi(1) = 10 \end{cases}$$

Alors, $\varphi(011) = \varphi(0)\varphi(1)\varphi(1) = 011010$

Théorèmes de clôture

Théorème : Clôture par morphisme

Soit $L \in$ **Reg** et φ un morphisme, alors $\varphi^{-1}(L) \in$ **Reg**.

Ce théorème peut être utilisé pour montrer qu'un langage n'est pas régulier.

Exemples

Soit $L_4 = \{w \in \{a, b\}^* | w \text{ a 2 fois plus de } a \text{ que de } b\}$

Théorèmes de clôture

Théorème : Clôture par morphisme

Soit $L \in$ **Reg** et φ un morphisme, alors $\varphi^{-1}(L) \in$ **Reg**.

Ce théorème peut être utilisé pour montrer qu'un langage n'est pas régulier.

Exemples

Soit $L_4 = \{w \in \{a, b\}^* | w \text{ a 2 fois plus de } a \text{ que de } b\}$

Si L_4 est régulier, alors $L_4 \cap a^*b^* = \{a^{2n}b^n | n \geq 0\}$ l'est aussi.
Or, soit $\varphi(0) = aa$ et $\varphi(1) = b$, alors $\varphi^{-1}(L_4 \cap a^*b^*) = \{0^n1^n | n \geq 0\}$ que l'on sait irrégulier.
Dès lors L_4 est irrégulier.