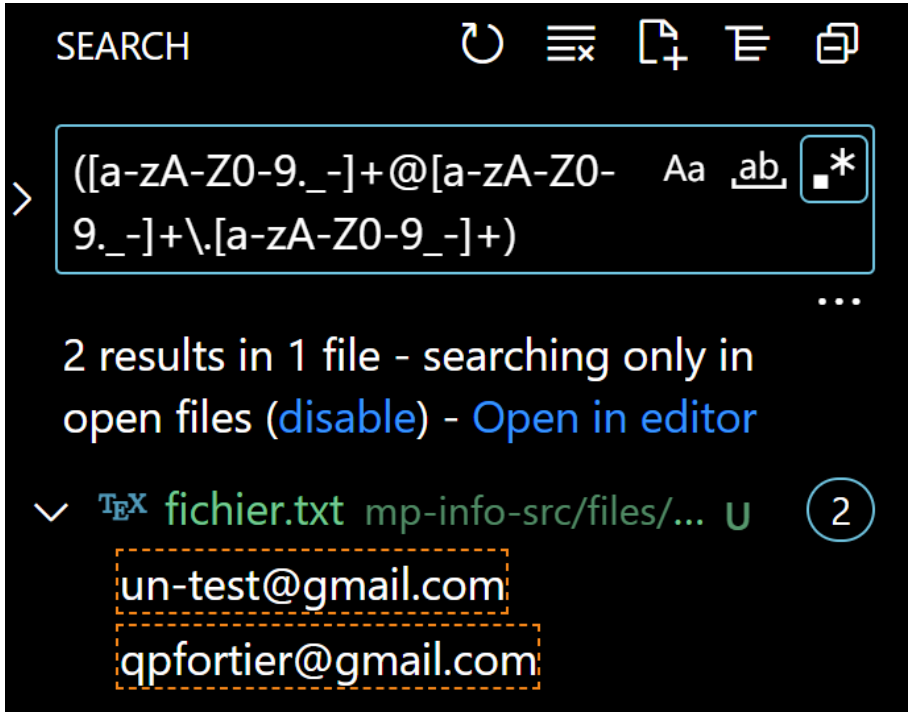
Langages réguliers

Quentin Fortier

September 7, 2026

Motivation des langages formels

- Recherche de motif dans un texte.



Recherche d'une adresse électronique dans Visual Studio Code

- Formalisation de la syntaxe d'un langage de programmation (et conception d'un nouveau langage) : BNF d'OCaml, de Python.

Mots

Définitions

- Un alphabet est un ensemble Σ fini, dont les éléments sont des lettres.

Mots

Définitions

- Un alphabet est un ensemble Σ fini, dont les éléments sont des lettres.
- Un mot m sur un alphabet Σ est une suite finie $m_1, \ldots, m_n$ de lettres de Σ, et on note $m = m_1 \cdots m_n$.

Mots

Définitions

- Un alphabet est un ensemble Σ fini, dont les éléments sont des lettres.
- Un mot m sur un alphabet Σ est une suite finie $m_1, \ldots, m_n$ de lettres de Σ, et on note $m = m_1 \cdots m_n$.
  n est la longueur de m, qu'on note $|m|$.

Mots

Définitions

- Un alphabet est un ensemble Σ fini, dont les éléments sont des lettres.
- Un mot m sur un alphabet Σ est une suite finie $m_1, \ldots, m_n$ de lettres de Σ, et on note $m = m_1 \cdots m_n$.
  n est la longueur de m, qu'on note $|m|$.
- Le mot vide (ne contenant aucune lettre) est noté ε.

Mots

Définitions

- Un alphabet est un ensemble Σ fini, dont les éléments sont des lettres.
- Un mot m sur un alphabet Σ est une suite finie $m_1, \ldots, m_n$ de lettres de Σ, et on note $m = m_1 \cdots m_n$.
  n est la longueur de m, qu'on note $|m|$.
- Le mot vide (ne contenant aucune lettre) est noté ε.
- On note Σ^* l'ensemble des mots sur Σ et $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$.

Mots

Définitions

- Un alphabet est un ensemble Σ fini, dont les éléments sont des lettres.
- Un mot m sur un alphabet Σ est une suite finie $m_1, \ldots, m_n$ de lettres de Σ, et on note $m = m_1 \cdots m_n$.
  n est la longueur de m, qu'on note $|m|$.
- Le mot vide (ne contenant aucune lettre) est noté ε.
- On note Σ^* l'ensemble des mots sur Σ et $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$.
- On note Σ^n l'ensemble des mots de longueur n sur Σ.

Mots

Définitions

- Un alphabet est un ensemble Σ fini, dont les éléments sont des lettres.
- Un mot m sur un alphabet Σ est une suite finie $m_1, \ldots, m_n$ de lettres de Σ, et on note $m = m_1 \cdots m_n$.
  n est la longueur de m, qu'on note $|m|$.
- Le mot vide (ne contenant aucune lettre) est noté ε.
- On note Σ^* l'ensemble des mots sur Σ et $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$.
- On note Σ^n l'ensemble des mots de longueur n sur Σ.

Remarque : on confond parfois la lettre a et le mot a (contenant une seule lettre).

Mots

Définition : Égalité de mots

Deux mots $u = u_1 \cdots u_n$ et $v = v_1 \cdots v_p$ sur le même alphabet Σ sont égaux s'ils ont la même longueur ($n = p$) et si, pour tout $i \in \{1, \ldots, n\}$, $u_i = v_i$.

Mots

Définition : Égalité de mots

Deux mots $u = u_1 \cdots u_n$ et $v = v_1 \cdots v_p$ sur le même alphabet Σ sont égaux s'ils ont la même longueur ($n = p$) et si, pour tout $i \in \{1, \ldots, n\}$, $u_i = v_i$.

Définition : Concaténation et puissance

- La concaténation de deux mots $u = u_1 \cdots u_n$ et $v = v_1 \cdots v_p$ est :

$$uv = u_1 \cdots u_n v_1 \cdots v_p$$

  Elle est aussi parfois notée $u \cdot v$.

Mots

Définition : Égalité de mots

Deux mots $u = u_1 \cdots u_n$ et $v = v_1 \cdots v_p$ sur le même alphabet Σ sont égaux s'ils ont la même longueur ($n = p$) et si, pour tout $i \in \{1, \ldots, n\}$, $u_i = v_i$.

Définition : Concaténation et puissance

- La concaténation de deux mots $u = u_1 \cdots u_n$ et $v = v_1 \cdots v_p$ est :
$$uv = u_1 \cdots u_n v_1 \cdots v_p$$
Elle est aussi parfois notée $u \cdot v$.
- Si u est un mot, on définit $u^0 = \varepsilon$ et $u^k = \underbrace{u \cdots u}_{k \text{ facteurs}}$.

Mots

Définition : Égalité de mots

Deux mots $u = u_1 \cdots u_n$ et $v = v_1 \cdots v_p$ sur le même alphabet Σ sont égaux s'ils ont la même longueur ($n = p$) et si, pour tout $i \in \{1, \ldots, n\}$, $u_i = v_i$.

Définition : Concaténation et puissance

- La concaténation de deux mots $u = u_1 \cdots u_n$ et $v = v_1 \cdots v_p$ est :
$$uv = u_1 \cdots u_n v_1 \cdots v_p$$
Elle est aussi parfois notée $u \cdot v$.
- Si u est un mot, on définit $u^0 = \varepsilon$ et $u^k = \underbrace{u \cdots u}_{k \text{ facteurs}}$.

Exercice

Soient Σ un alphabet, a, $b \in \Sigma$ et $u \in \Sigma^*$. On suppose $au = ub$. Montrer que $a = b$ et qu'il existe $k \in \mathbb{N}$ tel que $u = a^k$.

Mots

Un monoïde (HP) est comme un groupe, sauf qu'il n'y a pas forcément d'inverse.

Lemme

$(\Sigma^*, \cdot)$ est un monoïde (où $\cdot$ est la concaténation de mots), c'est-à-dire :

- $\varepsilon \cdot m = m \cdot \varepsilon = m$ (ε est un élément neutre) ;
- $(m \cdot n) \cdot p = m \cdot (n \cdot p)$ (associativité).

Mots

Un monoïde (HP) est comme un groupe, sauf qu'il n'y a pas forcément d'inverse.

Lemme

$(\Sigma^*, \cdot)$ est un monoïde (où $\cdot$ est la concaténation de mots), c'est-à-dire :

- $\varepsilon \cdot m = m \cdot \varepsilon = m$ (ε est un élément neutre) ;
- $(m \cdot n) \cdot p = m \cdot (n \cdot p)$ (associativité).

Lemme

La fonction qui, à un mot, associe sa longueur est un morphisme de monoïde de $(\Sigma^*, \cdot)$ vers $(\mathbb{N}, +)$, c'est-à-dire :

- $|\varepsilon| = 0$;
- $|m \cdot n| = |m| + |n|$.

Mots

Soient u et m deux mots de Σ^*.

Définitions

- u est un préfixe de m s'il existe un mot v tel que $m = uv$.

Mots

Soient u et m deux mots de Σ^*.

Définitions

- u est un préfixe de m s'il existe un mot v tel que $m = uv$.
- u est un suffixe de m s'il existe un mot v tel que $m = vu$.

Mots

Soient u et m deux mots de Σ^*.

Définitions

- u est un préfixe de m s'il existe un mot v tel que $m = uv$.
- u est un suffixe de m s'il existe un mot v tel que $m = vu$.
- u est un facteur (*substring* en anglais) de m s'il existe des mots v, w tels que $m = vuw$.

Mots

Soient u et m deux mots de Σ^*.

Définitions

- u est un préfixe de m s'il existe un mot v tel que $m = uv$.
- u est un suffixe de m s'il existe un mot v tel que $m = vu$.
- u est un facteur (*substring* en anglais) de m s'il existe des mots v, w tels que $m = vuw$.
- u est un sous-mot (*subsequence* en anglais) de m si u est une sous-suite (ou : suite extraite) de m.

Exemple : abc est un sous-mot de $aabacb$, mais pas un facteur.

Mots

Soient u et m deux mots de Σ^*.

Définitions

- u est un préfixe de m s'il existe un mot v tel que $m = uv$.
- u est un suffixe de m s'il existe un mot v tel que $m = vu$.
- u est un facteur (*substring* en anglais) de m s'il existe des mots v, w tels que $m = vuw$.
- u est un sous-mot (*subsequence* en anglais) de m si u est une sous-suite (ou : suite extraite) de m.

Exemple : abc est un sous-mot de $aabacb$, mais pas un facteur.

Exercice

Soit m un mot de longueur n dont toutes les lettres sont différentes.
Quel est son nombre de préfixes, de suffixes, de facteurs et de sous-mots ?
Et si des lettres peuvent être répétées ?

Mots

Rappels d'utilisation d'une chaîne de caractères (**string**) :

```
let s = "abc" (* définit une chaîne de caractères *)
s.[1] (* donne 'b' *)
String.length s (* donne 3 *)
"abc" ^ "def" (* concaténation *)
```

Remarque : contrairement à **array**, le type **string** est immuable (on ne peut pas modifier un caractère d'une chaîne de caractères).

Mots

Rappels d'utilisation d'une chaîne de caractères (`string`) :

```
let s = "abc" (* définit une chaîne de caractères *)
s.[1] (* donne 'b' *)
String.length s (* donne 3 *)
"abc" ^ "def" (* concaténation *)
```

Remarque : contrairement à `array`, le type `string` est immuable (on ne peut pas modifier un caractère d'une chaîne de caractères).

Question

Écrire une fonction `sous_mot : string -> string -> bool` qui teste si un mot est un sous-mot d'un autre, en complexité linéaire.

Langages

Définition

Un langage L sur un alphabet Σ est un ensemble de mots sur Σ.

De façon équivalente, L est un langage si $L \subseteq \Sigma^*$ ou encore $L \in \mathcal{P}(\Sigma^*)$.

Langages

Définition

Un langage L sur un alphabet Σ est un ensemble de mots sur Σ.

De façon équivalente, L est un langage si $L \subseteq \Sigma^*$ ou encore $L \in \mathcal{P}(\Sigma^*)$.

Exemples :

1. L'ensemble $L_0 = \{\varepsilon, a, bab\}$ sur $\Sigma = \{a, b\}$.
2. L'ensemble L_1 des mots d'un dictionnaire ne contenant que des lettres minuscules non accentuées, sur $\Sigma = \{a, b, \ldots, z\}$.
3. L'ensemble L_2 des formules arithmétiques sur $\Sigma = \{0, \ldots, 9, +, -, /, *, (,)\}$.
4. L'ensemble L_3 des programmes OCaml sur $\Sigma = \{a, \ldots, z, \_, !, <, >, \ldots\}$.

Langages

Exercice : Résumé des définitions

Soit $\Sigma = \{a, b\}$.

1. Σ est
2. a est
3. ε est
4. $abaaabb$ est
5. $\{a, b, abaaabb\}$ est

Langages

Exercice : Résumé des définitions

Soit $\Sigma = \{a, b\}$.

1. Σ est un alphabet.
2. a est une lettre (et aussi un mot de longueur 1).
3. ε est le mot vide.
4. $abaaabb$ est un mot.
5. $\{a, b, abaaabb\}$ est un langage.

Langages

On veut des algorithmes pour résoudre les problèmes suivants :

Problème

Étant donné un langage L et un mot m, est-ce que $m \in L$?

Applications : correcteur orthographique, coloration syntaxique, etc.

Langages

On veut des algorithmes pour résoudre les problèmes suivants :

Problème

Étant donné un langage L et un mot m, est-ce que $m \in L$?

Applications : correcteur orthographique, coloration syntaxique, etc.

Problème

Étant donné un texte s et un langage L, s contient-il un mot de L ?

Langages

On veut des algorithmes pour résoudre les problèmes suivants :

Problème

Étant donné un langage L et un mot m, est-ce que $m \in L$?

Applications : correcteur orthographique, coloration syntaxique, etc.

Problème

Étant donné un texte s et un langage L, s contient-il un mot de L ?
(Cas particulier $L = \{m\}$: le mot m apparaît-il dans un texte s?)

Application : recherche de motif (adresse électronique, séquence d'ADN, etc.) dans un texte.

Opérations sur les langages

Soient L_1 et L_2 deux langages sur le même alphabet.

Définition : Concaténation

La concaténation $L_1 L_2$ de L_1 et L_2 est définie par :

$$L_1 L_2 = \{m_1 m_2 \mid m_1 \in L_1,\ m_2 \in L_2\}$$

Opérations sur les langages

Soient L_1 et L_2 deux langages sur le même alphabet.

Définition : Concaténation

La concaténation $L_1 L_2$ de L_1 et L_2 est définie par :

$$L_1 L_2 = \{m_1 m_2 \mid m_1 \in L_1,\ m_2 \in L_2\}$$

$L_1 L_2$ est donc l'ensemble des mots obtenus par concaténation d'un mot de L_1 et d'un mot de L_2.

Opérations sur les langages

Soient L_1 et L_2 deux langages sur le même alphabet.

Définition : Concaténation

La concaténation $L_1 L_2$ de L_1 et L_2 est définie par :

$$L_1 L_2 = \{m_1 m_2 \mid m_1 \in L_1,\ m_2 \in L_2\}$$

$L_1 L_2$ est donc l'ensemble des mots obtenus par concaténation d'un mot de L_1 et d'un mot de L_2.

Exercice

1. Soit $L_1 = \{a, ab\}$ et $L_2 = \{\varepsilon, b, bba\}$. Déterminer $L_1 L_2$.
2. Quel lien a-t-on entre $|L_1 L_2|$ et $|L_1||L_2|$, dans le cas général ?

Opérations sur les langages

Exercice

1. La concaténation de deux langages est-elle commutative ? Associative ?
2. Quel est l'élément neutre de la concaténation ? L'élément absorbant ?

Opérations sur les langages

Exercice

1. La concaténation de deux langages est-elle commutative ? Associative ?
2. Quel est l'élément neutre de la concaténation ? L'élément absorbant ?

La concaténation est distributive par rapport à l'union :
$L_1(L_2 \cup L_3) = L_1 L_2 \cup L_1 L_3$.

Opérations sur les langages

Définition : Puissance

On définit la puissance L^n du langage L par récurrence :

$$L^0 = \{\varepsilon\}$$
$$L^n = L^{n-1}L, \text{ pour } n \geq 1$$

Opérations sur les langages

Définition : Puissance

On définit la puissance L^n du langage L par récurrence :

$$L^0 = \{\varepsilon\}$$
$$L^n = L^{n-1}L, \text{ pour } n \geq 1$$

Dit autrement : $L^n = \underbrace{L \cdot \ldots \cdot L}_{n \text{ facteurs}} = \{m_1 \cdots m_n \mid m_1 \in L, \ldots, m_n \in L\}$.

Opérations sur les langages

Définition : Puissance

On définit la puissance L^n du langage L par récurrence :

$$L^0 = \{\varepsilon\}$$
$$L^n = L^{n-1}L, \text{ pour } n \geq 1$$

Dit autrement : $L^n = \underbrace{L \cdot \ldots \cdot L}_{n \text{ facteurs}} = \{m_1 \cdots m_n \mid m_1 \in L, \ldots, m_n \in L\}$.

Exemple : Σ^n est l'ensemble des mots de longueur n sur l'alphabet Σ.

Opérations sur les langages

Définition : Puissance

On définit la puissance L^n du langage L par récurrence :

$$L^0 = \{\varepsilon\}$$
$$L^n = L^{n-1}L, \text{ pour } n \geq 1$$

Dit autrement : $L^n = \underbrace{L \cdot \ldots \cdot L}_{n \text{ facteurs}} = \{m_1 \cdots m_n \mid m_1 \in L, \ldots, m_n \in L\}$.

Exemple : Σ^n est l'ensemble des mots de longueur n sur l'alphabet Σ.

Exercice

1. Montrer que $L \subseteq L^2$ si et seulement si $L = \emptyset$ ou $\varepsilon \in L$.
2. Comparer L^2 et $\{u^2 \mid u \in L\}$.

Opérations sur les langages

Soit L un langage.

Définition : Étoile de Kleene

On définit l'étoile (de Kleene) L^* d'un langage L par :

$$L^* = \bigcup_{k \in \mathbb{N}} L^k$$

Opérations sur les langages

Soit L un langage.

Définition : Étoile de Kleene

On définit l'étoile (de Kleene) L^* d'un langage L par :

$$L^* = \bigcup_{k \in \mathbb{N}} L^k$$

L^* est donc l'ensemble des mots obtenus par concaténation d'un nombre quelconque de mots de L.

Opérations sur les langages

Soit L un langage.

Définition : Étoile de Kleene

On définit l'étoile (de Kleene) L^* d'un langage L par :

$$L^* = \bigcup_{k \in \mathbb{N}} L^k$$

L^* est donc l'ensemble des mots obtenus par concaténation d'un nombre quelconque de mots de L.
Remarque : L^* contient toujours ε, car $L^0 = \{\varepsilon\}$.

Opérations sur les langages

Soit L un langage.

Définition : Étoile de Kleene

On définit l'étoile (de Kleene) L^* d'un langage L par :

$$L^* = \bigcup_{k \in \mathbb{N}} L^k$$

L^* est donc l'ensemble des mots obtenus par concaténation d'un nombre quelconque de mots de L.
Remarque : L^* contient toujours ε, car $L^0 = \{\varepsilon\}$.

Question

Montrer que $(L^*)^* = L^*$.

Langages réguliers

Définition

L'ensemble des langages réguliers sur Σ est le plus petit ensemble (pour $\subset$) contenant les langages finis sur Σ et stable par union, concaténation et étoile de Kleene.

Langages réguliers

Définition

L'ensemble des langages réguliers sur Σ est le plus petit ensemble (pour $\subset$) contenant les langages finis sur Σ et stable par union, concaténation et étoile de Kleene.

On a donc :

Propriété

- Tout langage fini est régulier.
- Si L_1 et L_2 sont réguliers, alors $L_1 \cup L_2$ est régulier.
- Si L_1 et L_2 sont réguliers, alors $L_1 L_2$ est régulier.
- Si L est régulier, alors L^* est régulier.

Langages réguliers

Propriété

Par récurrence immédiate, si $L_1, \ldots, L_n$ sont réguliers, alors $L_1 \cup \ldots \cup L_n$ et $L_1 L_2 \cdots L_n$ sont réguliers.

Attention :

- Une union infinie de langages réguliers n'est pas forcément régulière.
- Un langage particulier n'est pas forcément stable par concaténation.

Langages réguliers

Définition

- Tout langage fini est régulier.
- L_1 et L_2 réguliers $\implies$ $L_1 \cup L_2$ régulier.
- L_1 et L_2 réguliers $\implies$ $L_1 L_2$ régulier.
- L régulier $\implies$ L^* régulier.

Exemples :

1. Soit m un mot. Alors $\{m\}$ est fini, donc régulier ; on le note aussi m par abus de notation.

Langages réguliers

Définition

- Tout langage fini est régulier.
- L_1 et L_2 réguliers $\Longrightarrow$ $L_1 \cup L_2$ régulier.
- L_1 et L_2 réguliers $\Longrightarrow$ $L_1 L_2$ régulier.
- L régulier $\Longrightarrow$ L^* régulier.

Exemples :

1. Soit m un mot. Alors $\{m\}$ est fini, donc régulier ; on le note aussi m par abus de notation.
2. Σ est fini, donc régulier. Par conséquent, Σ^* est également régulier.

Langages réguliers

Définition

- Tout langage fini est régulier.
- L_1 et L_2 réguliers $\implies$ $L_1 \cup L_2$ régulier.
- L_1 et L_2 réguliers $\implies$ $L_1 L_2$ régulier.
- L régulier $\implies$ L^* régulier.

Exemples :

1. Soit m un mot. Alors $\{m\}$ est fini, donc régulier ; on le note aussi m par abus de notation.
2. Σ est fini, donc régulier. Par conséquent, Σ^* est également régulier.
3. Soit m un mot. L'ensemble des mots ayant m comme facteur

Langages réguliers

Définition

- Tout langage fini est régulier.
- L_1 et L_2 réguliers $\implies$ $L_1 \cup L_2$ régulier.
- L_1 et L_2 réguliers $\implies$ $L_1 L_2$ régulier.
- L régulier $\implies$ L^* régulier.

Exemples :

1. Soit m un mot. Alors $\{m\}$ est fini, donc régulier ; on le note aussi m par abus de notation.
2. Σ est fini, donc régulier. Par conséquent, Σ^* est également régulier.
3. Soit m un mot. L'ensemble des mots ayant m comme facteur est égal à $\Sigma^* m \Sigma^*$; c'est donc un langage régulier.
4. Soit $m = m_1 \cdots m_n$ un mot. L'ensemble des mots ayant m comme sous-mot

Langages réguliers

Définition

- Tout langage fini est régulier.
- L_1 et L_2 réguliers $\implies$ $L_1 \cup L_2$ régulier.
- L_1 et L_2 réguliers $\implies$ $L_1 L_2$ régulier.
- L régulier $\implies$ L^* régulier.

Exemples :

1. Soit m un mot. Alors $\{m\}$ est fini, donc régulier ; on le note aussi m par abus de notation.
2. Σ est fini, donc régulier. Par conséquent, Σ^* est également régulier.
3. Soit m un mot. L'ensemble des mots ayant m comme facteur est égal à $\Sigma^* m \Sigma^*$; c'est donc un langage régulier.
4. Soit $m = m_1 \cdots m_n$ un mot. L'ensemble des mots ayant m comme sous-mot est égal à $\Sigma^* m_1 \Sigma^* m_2 \cdots \Sigma^* m_n \Sigma^*$; c'est donc un langage régulier.

Langages réguliers

Exercice

Montrer que les langages suivants sont réguliers sur $\Sigma = \{a, b\}$:

1. Mots commençant par a.
2. Mots commençant par a et finissant par b.
3. Mots de longueur paire.
4. Mots de longueur impaire.

Expressions régulières

Les expressions régulières sont une notation plus concise pour représenter un langage régulier :

Expressions régulières

L'ensemble $\mathcal{R}(\Sigma)$ des expressions régulières sur un alphabet Σ est le plus petit ensemble vérifiant :

- $\forall a \in \Sigma,\ a \in \mathcal{R}(\Sigma)$
- $\emptyset \in \mathcal{R}(\Sigma),\ \varepsilon \in \mathcal{R}(\Sigma)$
- $\forall e_1, e_2 \in \mathcal{R}(\Sigma),\ (e_1|e_2) \in \mathcal{R}(\Sigma)$ et $(e_1\, e_2) \in \mathcal{R}(\Sigma)$
- $\forall e \in \mathcal{R}(\Sigma),\ e^* \in \mathcal{R}(\Sigma)$

Expressions régulières

Les expressions régulières sont une notation plus concise pour représenter un langage régulier :

Expressions régulières

L'ensemble $\mathcal{R}(\Sigma)$ des expressions régulières sur un alphabet Σ est le plus petit ensemble vérifiant :

- $\forall a \in \Sigma,\ a \in \mathcal{R}(\Sigma)$
- $\emptyset \in \mathcal{R}(\Sigma),\ \varepsilon \in \mathcal{R}(\Sigma)$
- $\forall e_1, e_2 \in \mathcal{R}(\Sigma),\ (e_1|e_2) \in \mathcal{R}(\Sigma)$ et $(e_1\, e_2) \in \mathcal{R}(\Sigma)$
- $\forall e \in \mathcal{R}(\Sigma),\ e^* \in \mathcal{R}(\Sigma)$

Remarques :

- On utilisera seulement les parenthèses nécessaires. Ainsi, $(((ab)c)|d)$ sera noté $abc|d$.
- $e_1|e_2$ est aussi noté $e_1 + e_2$.

Expressions régulières

Les expressions régulières sont une notation plus concise pour représenter un langage régulier :

Expressions régulières

L'ensemble $\mathcal{R}(\Sigma)$ des expressions régulières sur un alphabet Σ est le plus petit ensemble vérifiant :

- $\forall a \in \Sigma,\ a \in \mathcal{R}(\Sigma)$
- $\emptyset \in \mathcal{R}(\Sigma),\ \varepsilon \in \mathcal{R}(\Sigma)$
- $\forall e_1, e_2 \in \mathcal{R}(\Sigma),\ (e_1|e_2) \in \mathcal{R}(\Sigma)$ et $(e_1\, e_2) \in \mathcal{R}(\Sigma)$
- $\forall e \in \mathcal{R}(\Sigma),\ e^* \in \mathcal{R}(\Sigma)$

Remarques :

- On utilisera seulement les parenthèses nécessaires. Ainsi, $(((ab)c)|d)$ sera noté $abc|d$.
- $e_1|e_2$ est aussi noté $e_1 + e_2$.

Exemples : a^*, $(a|aba)^*$, $a(a|b)^*b$ sont des expressions régulières.

Expressions régulières

Expressions régulières

L'ensemble $\mathcal{R}(\Sigma)$ des expressions régulières sur un alphabet Σ est le plus petit ensemble vérifiant :

- $\forall a \in \Sigma,\ a \in \mathcal{R}(\Sigma)$
- $\emptyset \in \mathcal{R}(\Sigma),\ \varepsilon \in \mathcal{R}(\Sigma)$
- $\forall e_1, e_2 \in \mathcal{R}(\Sigma),\ (e_1|e_2) \in \mathcal{R}(\Sigma)$ et $(e_1\, e_2) \in \mathcal{R}(\Sigma)$
- $\forall e \in \mathcal{R}(\Sigma),\ e^* \in \mathcal{R}(\Sigma)$

```
type 'a regexp =
    | Vide | Epsilon | L of 'a (* L a est la lettre a *)
    | Union of 'a regexp * 'a regexp
    | Concat of 'a regexp * 'a regexp
    | Etoile of 'a regexp
```

Expressions régulières

Expressions régulières

L'ensemble $\mathcal{R}(\Sigma)$ des expressions régulières sur un alphabet Σ est le plus petit ensemble vérifiant :

- $\forall a \in \Sigma,\ a \in \mathcal{R}(\Sigma)$
- $\emptyset \in \mathcal{R}(\Sigma),\ \varepsilon \in \mathcal{R}(\Sigma)$
- $\forall e_1, e_2 \in \mathcal{R}(\Sigma),\ (e_1|e_2) \in \mathcal{R}(\Sigma)$ et $(e_1\, e_2) \in \mathcal{R}(\Sigma)$
- $\forall e \in \mathcal{R}(\Sigma),\ e^* \in \mathcal{R}(\Sigma)$

```
type 'a regexp =
    | Vide | Epsilon | L of 'a (* L a est la lettre a *)
    | Union of 'a regexp * 'a regexp
    | Concat of 'a regexp * 'a regexp
    | Etoile of 'a regexp
```

Par exemple, $a(a|b)^*b$ est représenté par :

```
Concat(L 'a', Concat(Etoile(Union(L 'a', L 'b')), L 'b'))
```

Expressions régulières

```
type 'a regexp =
    | Vide | Epsilon | L of 'a
    | Union of 'a regexp * 'a regexp
    | Concat of 'a regexp * 'a regexp
    | Etoile of 'a regexp
```

Question

Écrire une fonction `lettres : 'a regexp -> 'a list` qui renvoie la liste des lettres utilisées dans une expression régulière.

Expressions régulières

Langage d'une expression régulière

Le langage $L(e)$ d'une expression régulière e est défini récursivement :

- $L(a) = \{a\}$ si $a \in \Sigma$
- $L(\emptyset) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$
- $L(e|e') = L(e) \cup L(e')$
- $L(ee') = L(e)L(e')$
- $L(e^*) = L(e)^*$

Par abus de notation, on oublie souvent le $L(e)$ et on confond expression régulière et langage associé.

Expressions régulières

Langage d'une expression régulière

Le langage $L(e)$ d'une expression régulière e est défini récursivement :

- $L(a) = \{a\}$ si $a \in \Sigma$
- $L(\emptyset) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$
- $L(e|e') = L(e) \cup L(e')$
- $L(ee') = L(e)L(e')$
- $L(e^*) = L(e)^*$

Par abus de notation, on oublie souvent le $L(e)$ et on confond expression régulière et langage associé.

Équivalence entre langage régulier et expression régulière

Un langage L est régulier.

$\Longleftrightarrow$

Il existe une expression régulière e telle que $L = L(e)$.

Expressions régulières

Langage d'une expression régulière

Le langage $L(e)$ d'une expression régulière e est défini récursivement :

- $L(a) = \{a\}$ si $a \in \Sigma$
- $L(\emptyset) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$
- $L(e|e') = L(e) \cup L(e')$
- $L(ee') = L(e)L(e')$
- $L(e^*) = L(e)^*$

Exemples avec $\Sigma = \{a, b\}$:

- $(a|b)^*$: ensemble de tous les mots ($= \Sigma^*$).

Expressions régulières

Langage d'une expression régulière

Le langage $L(e)$ d'une expression régulière e est défini récursivement :

- $L(a) = \{a\}$ si $a \in \Sigma$
- $L(\emptyset) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$
- $L(e|e') = L(e) \cup L(e')$
- $L(ee') = L(e)L(e')$
- $L(e^*) = L(e)^*$

Exemples avec $\Sigma = \{a, b\}$:

- $(a|b)^*$: ensemble de tous les mots ($= \Sigma^*$).
- $(a|b)^*bb$:

Expressions régulières

Langage d'une expression régulière

Le langage $L(e)$ d'une expression régulière e est défini récursivement :

- $L(a) = \{a\}$ si $a \in \Sigma$
- $L(\emptyset) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$
- $L(e|e') = L(e) \cup L(e')$
- $L(ee') = L(e)L(e')$
- $L(e^*) = L(e)^*$

Exemples avec $\Sigma = \{a, b\}$:

- $(a|b)^*$: ensemble de tous les mots ($= \Sigma^*$).
- $(a|b)^*bb$: mots finissant par bb.

Expressions régulières

Exercice

Donner une expression régulière pour les langages suivants, sur $\Sigma = \{a, b\}$:

1. Mots contenant au plus un a.
2. Mots dont la longueur est congrue à 1 modulo 3.
3. Mots contenant un nombre pair de a.
4. Mots contenant un nombre impair de a.
5. Écritures en base 2 des entiers divisibles par 4.

Expressions régulières

Équivalence d'expressions régulières

Deux expressions régulières e_1 et e_2 sont dites équivalentes, ce que l'on note $e_1 \equiv e_2$, si elles définissent le même langage, c'est-à-dire si $L(e_1) = L(e_2)$.

Exemple : $(ab)^*a \equiv a(ba)^*$.

Expressions régulières

Propriétés sur les expressions régulières

Pour des expressions régulières e, e_1, e_2 et e_3 :

1. $e\emptyset \equiv$
2. $e\varepsilon \equiv$
3. $e|\emptyset \equiv$
4. $\emptyset^* \equiv$
5. $\varepsilon^* \equiv$
6. $(e_1|e_2)e_3 \equiv$
7. $e_1(e_2\,e_3) \equiv$

Induction structurelle

Théorème

Soit $\mathcal{P}(L)$ une propriété sur les langages réguliers L telle que :

- $\mathcal{P}(L)$ est vraie pour les langages L finis (cas de base)
- $\mathcal{P}(L_1) \wedge \mathcal{P}(L_2) \implies \mathcal{P}(L_1 L_2)$
- $\mathcal{P}(L_1) \wedge \mathcal{P}(L_2) \implies \mathcal{P}(L_1 \cup L_2)$
- $\mathcal{P}(L) \implies \mathcal{P}(L^*)$

Alors $\mathcal{P}(L)$ est vraie pour tout langage régulier L.

Induction structurelle

Théorème

Soit $\mathcal{P}(L)$ une propriété sur les langages réguliers L telle que :

- $\mathcal{P}(L)$ est vraie pour les langages L finis (cas de base)
- $\mathcal{P}(L_1) \wedge \mathcal{P}(L_2) \implies \mathcal{P}(L_1 L_2)$
- $\mathcal{P}(L_1) \wedge \mathcal{P}(L_2) \implies \mathcal{P}(L_1 \cup L_2)$
- $\mathcal{P}(L) \implies \mathcal{P}(L^*)$

Alors $\mathcal{P}(L)$ est vraie pour tout langage régulier L.

$\{L \mid \mathcal{P}(L)\}$ est alors un ensemble contenant les langages finis et stable par union, concaténation et étoile de Kleene.
Donc il contient tous les langages réguliers, par définition.

Induction structurelle

Méthode

De même, on peut démontrer une propriété $\mathcal{P}(e)$ sur les expressions régulières e en montrant :

- $\mathcal{P}(\emptyset)$, $\mathcal{P}(\varepsilon)$ sont vraies (cas de base)
- $\mathcal{P}(a)$ est vraie pour $a \in \Sigma$ (cas de base)
- $\mathcal{P}(e_1) \wedge \mathcal{P}(e_2) \implies \mathcal{P}(e_1 e_2)$
- $\mathcal{P}(e_1) \wedge \mathcal{P}(e_2) \implies \mathcal{P}(e_1 | e_2)$
- $\mathcal{P}(e) \implies \mathcal{P}(e^*)$

Induction structurelle

Exercice : Miroir

Si $m = m_1 \cdots m_n$ est un mot, on définit son miroir $\widetilde{m} = m_n \cdots m_1$.
Si L est un langage, on définit son miroir $\widetilde{L} = \{\widetilde{m} \mid m \in L\}$.

1. Donner une expression régulière du miroir de $a(a|b)^*b$.
2. Soit e une expression régulière de langage L. Montrer que $\widetilde{L}$ est régulier.
3. Écrire une fonction OCaml `miroir : 'a regexp -> 'a regexp` renvoyant le miroir d'une expression régulière.

Induction structurelle

Méthode

- Pour montrer une égalité de deux mots, on peut faire une récurrence sur la longueur du mot.
- Pour montrer une égalité de deux langages, on peut montrer une double inclusion.

Question

Soient e_1 et e_2 deux expressions régulières.
Montrer que $(e_1^* e_2)^* e_1^* \equiv (e_1 | e_2)^*$.

Expressions régulières en pratique (non exigible)

`grep -E` est une commande Linux qui permet de chercher des motifs dans un texte en utilisant des expressions régulières étendues :

Expression régulière	Signification
`.`	n'importe quel caractère
`[aei]`	un caractère parmi `a`, `e`, `i`
`[a-z]`	un caractère entre `a` et `z`
`[^aei]`	un caractère qui n'est pas `a`, `e`, `i`
⋮	⋮

Complément : Dénombrabilité et langages non réguliers

Soit Σ un alphabet non vide.

Théorème

Σ^+ (et donc aussi Σ^*) est infini dénombrable.

Complément : Dénombrabilité et langages non réguliers

Soit Σ un alphabet non vide.

Théorème

Σ^+ (et donc aussi Σ^*) est infini dénombrable.

Preuve : on identifie les p lettres de Σ avec les entiers de 1 à p.
L'écriture en base $p+1$ donne une injection de Σ^+ vers $\mathbb{N}$.

Complément : Dénombrabilité et langages non réguliers

Soit Σ un alphabet non vide.

Théorème

Σ^+ (et donc aussi Σ^*) est infini dénombrable.

Preuve : on identifie les p lettres de Σ avec les entiers de 1 à p.
L'écriture en base $p+1$ donne une injection de Σ^+ vers $\mathbb{N}$.

On en déduit :

Théorème

Un langage $L \subseteq \Sigma^*$ est au plus dénombrable.

Par exemple, le langage des programmes OCaml est dénombrable.

Complément : Dénombrabilité et langages non réguliers

Théorème de Cantor

Un ensemble E ne peut pas être en bijection avec $\mathcal{P}(E)$.

Complément : Dénombrabilité et langages non réguliers

Théorème de Cantor

Un ensemble E ne peut pas être en bijection avec $\mathcal{P}(E)$.

Preuve : si $f : E \longrightarrow \mathcal{P}(E)$ alors $Y = \{x \in E \mid x \notin f(x)\}$ n'a pas d'antécédent par f.

Complément : Dénombrabilité et langages non réguliers

Théorème de Cantor

Un ensemble E ne peut pas être en bijection avec $\mathcal{P}(E)$.

Preuve : si $f : E \longrightarrow \mathcal{P}(E)$ alors $Y = \{x \in E \mid x \notin f(x)\}$ n'a pas d'antécédent par f.

Corollaire

L'ensemble $\mathcal{P}(\Sigma^*)$ des langages sur Σ n'est pas dénombrable.

Complément : Dénombrabilité et langages non réguliers

Théorème de Cantor

Un ensemble E ne peut pas être en bijection avec $\mathcal{P}(E)$.

Preuve : si $f : E \longrightarrow \mathcal{P}(E)$ alors $Y = \{x \in E \mid x \notin f(x)\}$ n'a pas d'antécédent par f.

Corollaire

L'ensemble $\mathcal{P}(\Sigma^*)$ des langages sur Σ n'est pas dénombrable.

Comme l'ensemble des langages est indénombrable alors que l'ensemble des programmes OCaml est dénombrable, il existe des langages L pour lesquels le problème suivant ne peut pas être résolu par un algorithme :

Problème

Étant donné un mot m, est-ce que $m \in L$?

Complément : Dénombrabilité et langages non réguliers

Théorème de Cantor

Un ensemble E ne peut pas être en bijection avec $\mathcal{P}(E)$.

Preuve : si $f : E \longrightarrow \mathcal{P}(E)$ alors $Y = \{x \in E \mid x \notin f(x)\}$ n'a pas d'antécédent par f.

Corollaire

L'ensemble $\mathcal{P}(\Sigma^*)$ des langages sur Σ n'est pas dénombrable.

Comme l'ensemble des langages est indénombrable alors que l'ensemble des programmes OCaml est dénombrable, il existe des langages L pour lesquels le problème suivant ne peut pas être résolu par un algorithme :

Problème

Étant donné un mot m, est-ce que $m \in L$?

On va donc se restreindre à un ensemble plus simple de langages.

Complément : Dénombrabilité et langages non réguliers

Soit Σ un alphabet non vide.

Théorème

L'ensemble des langages réguliers sur Σ est infini dénombrable.

Preuve :

Complément : Dénombrabilité et langages non réguliers

Soit Σ un alphabet non vide.

Théorème

L'ensemble des langages réguliers sur Σ est infini dénombrable.

Preuve : l'ensemble des expressions régulières sur Σ est dénombrable (c'est un langage sur un alphabet fini), donc l'ensemble des langages réguliers est au plus dénombrable. De plus, si $a \in \Sigma$, les langages $\{a^n\}$, pour $n \in \mathbb{N}$, sont réguliers et deux à deux distincts.

Complément : Dénombrabilité et langages non réguliers

Soit Σ un alphabet non vide.

Théorème

L'ensemble des langages réguliers sur Σ est infini dénombrable.

Preuve : l'ensemble des expressions régulières sur Σ est dénombrable (c'est un langage sur un alphabet fini), donc l'ensemble des langages réguliers est au plus dénombrable. De plus, si $a \in \Sigma$, les langages $\{a^n\}$, pour $n \in \mathbb{N}$, sont réguliers et deux à deux distincts.

Comme l'ensemble de tous les langages sur Σ est indénombrable :

Corollaire

Il existe des langages non réguliers sur Σ.

Complément : Dénombrabilité et langages non réguliers

Soit Σ un alphabet non vide.

Théorème

L'ensemble des langages réguliers sur Σ est infini dénombrable.

Preuve : l'ensemble des expressions régulières sur Σ est dénombrable (c'est un langage sur un alphabet fini), donc l'ensemble des langages réguliers est au plus dénombrable. De plus, si $a \in \Sigma$, les langages $\{a^n\}$, pour $n \in \mathbb{N}$, sont réguliers et deux à deux distincts.

Comme l'ensemble de tous les langages sur Σ est indénombrable :

Corollaire

Il existe des langages non réguliers sur Σ.

On verra plus tard comment montrer qu'un langage n'est pas régulier.