**Langages réguliers, Automates d'états finis et Expresions régulières**

# 1 Langages réguliers

Soit $V$ un **alphabet**, c'est-à-dire un ensemble fini non vide de symboles. L'ensemble de tous les mots que l'on peut former avec des symboles de $V$ est noté $V^*$.
On appelle **langage** sur $V$ tout sous-ensemble de $V^*$.

L'ensemble des **langages réguliers** sur $V$ peut-être défini récursivement par les règles suivantes :
- le langage vide $\emptyset$ est un langage régulier ;
- le langage $\{\varepsilon\}$ est un langage régulier ;
- pour tout $x$ de $V$, $\{x\}$ est un langage régulier ;
- si $L1$ et $L2$ sont des langages réguliers alors : $L1 \cup L2$, $L1.L2$ et $L1^*$ sont des langages réguliers.

## Propriétés

$V^* \backslash L1$ et $L1 \cap L2$ sont des langages réguliers.

**Remarque**
Tous les langages ne sont pas réguliers !!! C'est par exemple le langage $\{a^n.b^n \mid n \geq 0\}$ défini sur le vocabulaire $\{a, b\}$. En particulier la plupart des langages de programmation ne sont pas des langages réguliers.

# 2 Automates d'états finis

Un intérêt des langages réguliers est que, pour tout langage régulier $L$ défini sur $V$, il existe un algorithme efficace pour décider si un mot quelconque de $V^*$ appartient ou non à $L$. Cet algorithme repose sur une caractérisation de $L$ à l'aide d'un automate d'états finis déterministe.

On appelle automate d'états finis un quintuplet $A = (Q, V, \delta, q_0, F)$ dans lequel :
- Q est un ensemble fini d'états ;
- V est un ensemble fini de symboles (le vocabulaire d'entrée, ou alphabet) ;
- $\delta \subseteq Q \times V \times Q$ est la relation de transition ;
- $q_0 \in Q$ est l'état initial ;
- $F \subseteq Q$ est l'ensemble des états terminaux.

Un mot $w = x_0.x_1.x_2.\ldots.x_{n-1}$ de $V^*$ est **accepté** par un automate $A$ si et seulement si il existe une séquence $q_0.q_1.q_2.\ldots.q_n$ de $Q^*$ telle que :
- pour tout $0 \leq i \leq n-1$, $(q_i, x_i, q_{i+1}) \in \delta$
- $q_n$ appartient à l'ensemble $F$

Le **langage reconnu** (ou accepté) par un automate $A$, noté $L(A)$ est constitué de l'ensemble des mots acceptés par $A$.

### Propriétés

— Pour tout langage régulier, il existe un automate d'états finis qui le reconnaît.
— Si $A$ est un automate d'états finis, $L(A)$ est un langage régulier.

### Opérations sur les automates

— **Déterminisation**
Un automate d'état fini $A$ est dit **déterministe** si et seulement si la relation de transition est une fonction (partielle), c'est-à-dire si elle vérifie la propriété suivante :

$$(\forall (p, q, q') \in Q^3).\ (\forall x \in V).\ \delta(p, x, q) \wedge \delta(p, x, q') \Longrightarrow q = q'$$

Si $A$ est un automate d'états finis, il existe un algorithme permettant de construire un automate déterministe reconnaisant le même langage.

— **Opérations de fermeture**
Soient $A1$ et $A2$ des automates d'états finis. Il existe des algorithmes permettant de construire des automates d'états finis reconnaissant :
— $L(A1) \cup L(A2)$
— $L(A1).L(A2)$
— $(L(A1)^*$
— $L(A1) \cap L(A2)$
— $V^* \backslash L(A1)$

— **Automate minimal**
Pour tout langage régulier, il existe un unique automate déterministe minimal (en nombre d'états) qui le reconnaît.
Pour tout automate d'états finis, il existe un algorithme calculant l'automate déterministe minimal qui le reconnît.

## 3 Application

L'intérêt des langages réguliers est qu'il existe un algorithme efficace permettant de décider si un mot fini $w$ appartient ou non à un langage régulier $L$ : il suffit en effet de construire un automate $A$ **déterministe** acceptant $L$ et de vérifier que cet automate accepte bien le mot $w$. L'automate $A$ étant déterministe, cette vérification peut se faire au fur et à mesure d'un parcours unique de $w$ (elle est donc linéaire en fonction de la taille de ce mot).

L'application principale est l'**analyse lexicale**, qui consiste à extraire une séquence de *lexèmes* (ou "mots") à partir d'une séquence de symboles (généralement des caractères). L'ensemble des lexèmes est défini par un langage régulier, et on utilise alors un automate pour reconnaitre chaque sous-séquence correspondant à un lexème donné. Certains symboles sont alors souvent considérés comme des "séparateurs" (ils peuvent ou doivent apparaitre entre les lexèmes).

L'analyse lexicale est notamment l'une des premières phases d'un compilateur.

# 4 Expressions régulières

Tout langage régulier sur $V$ peut être décrit par une **expression régulière**, c'est-à-dire un terme construit sur l'alphabet $V \cup \{+, ., *, \underline{\varepsilon}, \underline{\emptyset}\}$ de la manière suivante :

- $\underline{\emptyset}$ décrit le langage vide $\emptyset$
- $\underline{\varepsilon}$ décrit le langage $\{\varepsilon\}$
- pour tout $x$ appartenant à $V$, $x$ décrit le langage $\{x\}$
- si $r1$ et $r2$ sont des expressions régulières sur $V$ décrivant respectivement les langages $L1$ et $L2$ alors :
  - $r1 + r2$ décrit le langage $L1 \cup L2$
  - $r1.r2$ décrit le langage $L1.L2$
  - $r1^*$ décrit le langage $L1^*$

On considérera dans la suite que l'opérateur unaire $*$ est plus prioritaire que l'opérateur ., lui-même plus prioritaire que l'opérateur $+$. Lorsque cela ne génère pas d'ambiguïté, le . est souvent omis. Par exemple, on écrira $ab$ au lieu de $a.b$.

**Exemples :**

| Expression régulière | Langage |
|---|---|
| $a + b.c$ | $\{a, bc\}$ |
| $(ab)^*$ | $\{\varepsilon, ab, abab, ababab, \ldots\}$ |
| $ab + c*$ | $\{\varepsilon, ab, c, cc, ccc, \ldots\}$ |
| $((b+c)^*a(b+c)^*a(b+c)^*)^*$ | mots ayant un nombre pair de $a$ |
| $(a+b+c)^*abc$ | mots se terminant par $abc$ |

## Propriétés

- Pour toute expression régulière, il existe un algorithme permettant de construire un automate d'états finis équivalent (cad qui reconnaît le même langage que celui décrit par l'expression régulière).

- Pour tout automate d'états finis, il existe un algorithme permettant de calculer une expression régulière équivalente.