## Langages formels

Alphabet, mot, opération sur les mots, langages,

Grammaires, arbres de dérivation.

Classification de Chomsky…

*Auteur :* Olivier Raynaud

Version de travail : rentrée 2020

Sensibilité :

*Référence :*

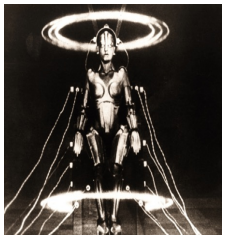
1

## Théorie des automates

❑ *Des années trente à nos jours…*

**Définition** intuitive (théorie des automates) **:** La théorie des automates est l'étude des machines abstraites. Elle traite et décrit précisément la frontière entre ce qu'une machine peut faire et ce qu'elle ne peut pas faire.

- **Années 30** Travaux de A. Turing, définition d'une machine abstraite définissant les capacités des machines 'aujourd'hui.
  *ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM*
- **Années 40 et 50** Automate à états finis pour la modélisation de fonctions cognitives puis exploités pour de nombreuses applications.
- **Années 50** Le linguiste N. Chomsky étudie les grammaires génératives, c.à.d.. les grammaires formelles, qui admettent des liens puissants avec la théories des automates.
- **Années 60** S. Cook prolonge les travaux de A. Turing et identifie les problèmes qui peuvent être traités en temps raisonnable de ceux qui ne le peuvent pas.

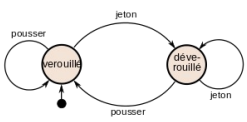


2

## Théorie des automates

❑ *Point de vue applicatif*

- **Conception logicielle** Outil de modélisation pour la conception et la vérification du comportement des circuits numériques
- **Compilation** Outil pour l'analyse lexicale de composant d'un compilateur qui décompose un programme en unités logiques tels que les identifieurs, les mots clés ou les symboles de ponctuation.
- **Traitement de texte** Outil de parcours de larges portions de textes telle que des collection de pages web, pour identifier les occurrences d'un mot, de phrases ou d'autres types de motifs.
- **Systèmes mécaniques** Outil de vérification de système de tous type composés d'un nombre fini d'états tels que protocoles de communication ou les protocoles d'échanges d'information sensibles.

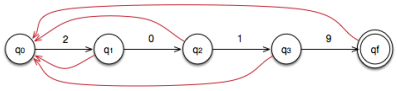



https://fr.wikipedia.org/wiki/Automate_fini

3

## Théorie des automates

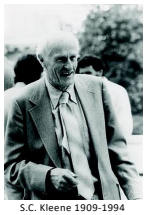
❑ *Acceptation d'une séquence de caractères*




- Le travail de cet automate est de reconnaître une séquence de chiffres comme le code d'un digicode par exemple.
- Nous avons 5 états, les entrées de l'automate sont les chiffres soumis par l'utilisateur externe au système. Chaque état représente une étape franchie dans la composition du code.
- A chaque erreur le système nous ramène au début nous permettant de recomposer une nouvelle séquence.

4

## Expressions régulières



S.C. Kleene 1909-1994

**Définition** intiutive (regex) : une **expression régulière** est une chaîne de caractères décrivant de façon concise un ensemble de mots sur un alphabet donné.

Elles sont exploitées pour implémenter des fonctions de lecture , de contrôle, de modification et d'analyse de textes.

$$([A-Z][a-z]^*[\ ])^*[\ ][A-Z][A-Z]$$

```
\w+([-+.']\w+)*@\w+([-.]\w+)*\.[a-zA-Z]{2,6}
```

5

## Alphabet

**Définition** (alphabet) :
On appelle **alphabet** et on note $\Sigma$ un ensemble fini de caractères appelés parfois symboles.

Dans la suite du cours, nous ne considérerons que des alphabets de taille fini.

Exemples :

$\Sigma = \{0, 1\}$, l'alphabet binaire ; $\Sigma = \{a, b, ..., z\}$, l'alphabet des lettres minuscules ;

L'ensemble des tous les caractères ASCII ou
de tous les caractères ASCII imprimables.

https://www.ionos.fr/digitalguide/serveur/know-how/ascii-american-standard-code-for-information-interchange/

6

## Mots

**Définition** intuitive (mot, longueur) :
On appelle **mot** sur $\Sigma$ (appelé aussi **chaîne**) et on note $w$ une suite finie de symboles de $\Sigma$ juxtaposés.
On appelle **longueur** d'un mot $w$ et on note $|w|$ le nombre de caractères de $w$.
Pout tout $s$ dans $\Sigma$, on note $|w|_s$ le nombre d'occurrences de $s$ dans $w$.

Exemple: Soit $w = aabaa$ un mot sur $\Sigma = \{a, b\}$, on a $|w| = 5$ et $|w|_a = 4$

**Définition** (mot vide) :
Le **mot vide** est le mot ne contenant aucun caractère, il est noté $\epsilon$.
Par convention la longueur du mot vide est nulle : $|\epsilon| = 0$.

7

## Puissance d'un alphabet

❑ *Notation exponentielle*
✓ Soit un alphabet, nous pouvons exprimer l'ensemble de tous les mots d'une certaine longueur construits sur cet alphabet.

**Définition** (puissance d'un alphabet):
Soit un alphabet $\Sigma$, nous définissons et notons $\Sigma^k$ comme l'ensemble de tous les mots de longueur $k$ tels que chacun des caractères du mot est dans $\Sigma$.

Note : Pour tout $\Sigma$, on a $\Sigma^0 = \{\epsilon\}$.

Exemple :

Soit $\Sigma = \{0, 1\}$, on a :
$\Sigma^0 = \{\epsilon\}$, $\Sigma^1 = \{0, 1\}$,
$\Sigma^2 = \{00, 01, 10, 11\}$. $\Sigma^3 = \{000, 001, 011, 010, 100, 101, 110, 111\}$.

8

## Fermeture de Kleene

**Définition** (fermeture de Kleene) :
Soit un alphabet $\Sigma$, on appelle **fermeture de Kleene** de $\Sigma$, notée $\Sigma^*$ l'ensemble de tous les mots construits sur cet alphabet, mot vide inclus.

$$\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$$

Exemple : Soit $\Sigma = \{a, b\}$, $\Sigma^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

✓ Parfois, nous désirons exclure le mot vide de l'ensemble des mots.
Nous notons $\Sigma^+$ l'ensemble de tous les mots non vides construits sur $\Sigma$.
Nous obtenons les deux équivalences suivantes :

1. $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots$ ;
2. $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$.

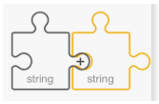
9

## Concaténation

**Définition** (concaténation – produit) :
Soient $\alpha = a_1 a_2 ... a_i$ et $\beta = b_1 b_2 ... b_j$ deux mots sur $\Sigma^*$, on appelle **concaténation** (ou **produit**) de $\alpha$ et $\beta$, le mot noté $\alpha.\beta = a_1 a_2 ... a_i b_1 b_2 ... b_j$.

Exemples:
Soient $\alpha = abba$ et $\beta = bb$ deux mots sur $\Sigma = \{a, b\}$,
$\alpha.\beta = abba.bb = abbabb$.

Pout tout mot $w$ on a : $\epsilon.w = w.\epsilon = w$.
$\epsilon$ est l'élément neutre de la concaténation.




Exemple :
Soient $\alpha = abba$ et $\beta = \epsilon$,
$\alpha.\beta = \beta.\alpha = abba.\epsilon = \epsilon.abba = abba$.

10

## Facteur

**Définition** (facteur et facteur propre) :
On appelle **facteur** d'un mot $w$, un mot $\alpha$ tel qu'il existe deux mots $\beta$ et $\gamma$ avec $w = \beta.\alpha.\gamma$. Un facteur est dit **propre** s'il est différent de $\epsilon$ et du mot $w$ lui même.

Exemples :
Soit $w = abbcca$ sur $\Sigma = \{a, b, c\}$,
le mot $cc$ est un facteur de $w$ car $w$ peut s'écrire $abb.cc.a$ ;
le mot $abbcca$ est un facteur non propre de $abbcca$.

http://villemin.gerard.free.fr/Wwwgvmm/Suite/Suitthue.htm

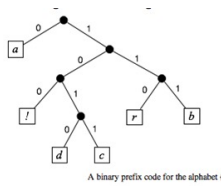
11

## Préfixe et suffixe

**Définition** (préfixe et suffixe) :
Soit un mot $w$, un mot $\alpha$ est **préfixe** de $w$, s'il existe un mot $\beta$ avec $w = \alpha.\beta$.
Un mot $\beta$ est **suffixe** de $w$, s'il existe un mot $\alpha$ avec $w = \alpha.\beta$.

Un préfixe ou un suffixe de $w$ est dit **propre** s'il est différent de $\epsilon$ et de $w$.

https://www.jeprevise.fr/prefixe-suffixe.html




| character | encoding |
|---|---|
| a | 0 |
| b | 111 |
| c | 1011 |
| d | 1010 |
| r | 110 |
| ! | 100 |

A binary prefix code for the alphabet {a, b, c, d, r, !}

https://www.chegg.com/homework-help/questions-and-answers/prefix-codes-prefix-codes-widely-used-applications-compress-data-including-jpeg-images-mp3-q1208636

12

## Langages

**Définition** (langage) :
Soit $\Sigma$ un alphabet, on appelle **langage** sur $\Sigma$ un ensemble de mots $L$ tel que $L \subseteq \Sigma^*$.

Exemples de langages abstraits :

Le langage des chaînes de bits contenant autant de 0 que de 1
$\{\epsilon, 01, 10, 0011, 0101, 1001, \ldots\}$

Le langage sur {0,1} des mots dont la valeur est nombre premier
$\{10, 11, 101, 111, 1011, \ldots\}$

Le langage vide, $\emptyset$ pour tout alphabet $\Sigma$.
Le langage $\{\epsilon\}$ ne contenant que le mot vide.


https://fr.123rf.com/photo_16720843_nuage-de-mot-abstrait-pour-langage-de-programmation-de-haut-niveau-avec-des-%C3%A9tiquettes-et-des-termes-c.html

13

## Le langage – quelques définitions historiques


Galilée 1564-1642

**Définition** : Suite d'habilétés permettant de relier le son au sens *(Aristote 380-320 av. J.-C.)*

**Définition** : Le *langage* est un ensemble de moyens fini pour exprimer un éventail illimité de pensées. *(Galilée 1564-1642)*

**Propriété fondamentale du langage** : Chaque langue fournit une palette infinie d'expressions structurées hiérarchiquement et qui peuvent être intérprétées au niveau de deux interfaces : l'interface conceptuelle intentionnelle et l'interface sensori motrice.

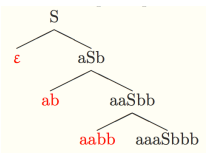

Chomsky 1928- …

14

## Définition d'un langage

- *Un langage peut être défini de plusieurs façons:*
- ✓ En **extension** : liste exhaustive de tous les mots du langage;
  - Exemple du dictionnaire:
- ✓ En **compréhension** : début d'une liste énumérative des mots du langage;
  - Exemple : L = {ab, aabb, aaabbb, …}
- ✓ En **intention** : énumération de règles en langage naturel ou formel.
  - Exemple : tous les mots formés de `a' et de `b' contenant autant des deux caractères et tels que les `a' sont placés avant les `b'.
- ✓ Par **induction** : utilisation d'outils compacts de description de langages

$G = \langle \{S\}, \{a, b\}, \{S \rightarrow aSb|\varepsilon\}, S \rangle$




15

## Définition ensembliste d'un langage

- *Un ensemble comme outil de **définition par intention***

✓ Il est commun de décrire un langage comme un ensemble sous la forme :

{ *w* | liste de propriétés concernant *w* }

cette définition peut être interprétée de la façon suivante:

... l'ensemble des mots *w* tel que les propriétés citées concernant *w* sont vraies.

*Exemples*

- { w | w contient autant de caractères `0' que de caractères `1'}
- { w | w est un programme en C syntaxiquement correct}

✓ On peut aussi décrire w par une expression paramétrée conditionnant les chaînes admises du langage.

*Exemples*

$\{0^n 1^n \mid n \leq 1\}$ $\{a^n b a^n \mid 0 \leq n\}$ $\{a^n b^n \mid 0 \leq n\}$ $\{0^i 1^j \mid 0 \leq i \leq j\}$

16

## Définition inductive d'un ensemble ou langage

- *Un ensemble peut être **défini inductivement** par un schéma en 3 étapes:*

➢ On fixe une base contenant les premiers éléments de l'ensemble à définir;

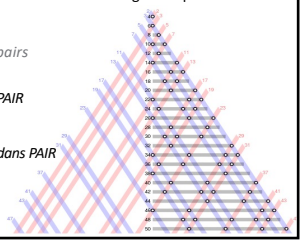
➢ On fixe des règles de production permettant de générer de nouveaux éléments à partir des éléments appartenant déjà à l'ensemble;

➢ On déclare que les seuls éléments de l'ensemble à définir sont ceux que l'on peut générer en appliquant un nombre de fois fini les règles de production.

*Exemples: Ensemble PAIR des nombres pairs*

*Base : PAIR = {0}*
*Règle : si x est dans PAIR alors x+2 est dans PAIR*

*Base : PAIR = {0,2}*
*Règle : si x et y sont dans PAIR alors x+y est dans PAIR*

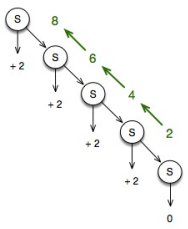


17

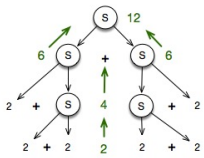
## Preuve d'appartenance

*Exemples: Ensemble PAIR des nombres pairs*

*Base : PAIR = {0}*
*Règle : si x est dans PAIR alors x+2 est dans PAIR*
**8 est-il pair?**




*Base : PAIR = {0,2}*
*Règle : si x et y sont dans PAIR alors x+y est dans PAIR*
**12 est-il pair?**




18

## Les grammaires

❑ *Les grammaires comme outils de définition d'un langages*

➢ La grammaire est un formalisme simple et compact de description d'un langage formel.

**Définition** intuition (Grammaire) **:** Une **grammaire** est conçue sous la forme d'un mécanisme génératif à l'aide d'un axiome de départ et d'un système de règles. La **grammaire** doit engendrer tous les mots possibles du langage qu'elle définit.

Extrait de [universalis.fr]

Puisque le langage est potentiellement infini, l'objet d'étude (le langage) ne peut être défini par un corpus observable, fini.

```
exp ::= exp + exp
      | exp × exp
      | (exp)
      | num
num ::= chiffre num
      | chiffre
chiffre ::=  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

19

## Grammaires

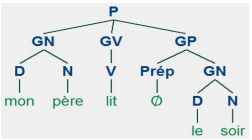
➢ **Introduite** par le langage Algol dans les années 60; Une grammaire est une technique simple et compacte de description de langages.

**Définition** formelle (Grammaire) **:** Une **grammaire formelle** ou simplement grammaire est un quadruplet **G = {N, T, P, S}** où

- V est un ensemble fini de symboles non terminaux ;
- T est un ensemble fini de terminaux;
- P est un ensemble fini de règles de production de la forme

$$\alpha \rightarrow \beta \text{ avec } \alpha \in (N \cup T)^+ \text{ et } \beta \in (N \cup T)^*$$

- S est un symbol de N appelé symbol de départ.




P --> GN.GV.GP  GP --> Prep.GN
GN --> D.N.  P --> GP  GV --> V

D --> mon|le  V --> lit
N --> père|soir  Prep --> eps

20

## Le langage des palindromes

❑ *Exemple de langage et de grammaire associée*

**Définition** (Palindrome) **:** Un **palindrome** est un mot, une phrase que l'on peut lire indifféremment de gauche à droite et de droite à gauche.

Quelques exemples :
Ressasser, Hannah, A cuba Anna a bu ça

Sur l'alphabet $\Sigma = \{0, 1\}$ nous avons
— base : $\epsilon, 0$ et $1$ sont des palindromes.
— induction : soit $w$ un palindrome
alors $0w0$ et $1w1$ sont des palindromes.

$G = \{S, \{0, 1\}, P, S\}$ avec

$P$ l'ensemble des règles :
$S \rightarrow 0 | \ 1 \ | \epsilon ;$  $S \rightarrow 0S0 \mid 1S1.$


Carré magique (sacré) Latin

21

## Relation de dérivation

**Définition** (Dérivation) :

Soit un grammaire $G = \{N, T, P, S\}$ alors on dit que
$\gamma$ se dérive en $\delta$ et on note $\gamma \Rightarrow \delta$
**si et seulement si**
$\exists \alpha \rightarrow \beta \in P$, et $\exists \gamma_1, \gamma_2 \in (N \cup T)^*$ tels que $\gamma = \gamma_1 \alpha \gamma_2 = \gamma_1 \beta \gamma_2 = \delta$

```
exp ::= exp + exp
      | exp × exp
      | (exp)
      | num
num ::= chiffre num
      | chiffre
chiffre ::=  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Chemin de dérivation

$exp \Rightarrow exp \times exp \Rightarrow num \times exp$
$\Rightarrow chiffre \times exp \Rightarrow 3 \times exp$
$\Rightarrow 3 \times num \Rightarrow 3 \times chiffre\ num$
$\Rightarrow 3 \times 1\ num \Rightarrow 3 \times 1\ chiffre$
$\Rightarrow 3 \times 18.$

**Note** : la relation $\overset{*}{\Rightarrow}$ est la fermeture reflexo-transitive de $\Rightarrow$.
**Exemple** : Nous avons $exp \times exp \overset{*}{\Rightarrow} 3 \times 1\ chiffre$.

22

## Définitions

**Définition** (Langage engendré par un grammaire) :
On appelle **langage engendré par une grammaire** G=(N,T,P,S), noté L(G), l'ensemble des mots w de T* que l'on peut obtenir par une suite finie de dérivation à partir de S.
Autrement dit nous avons :

$$L(G) = \{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$$

**Définition** (Equivalence de grammaires) :
Deux grammaires G et G' sont dites équivalentes si L(G) = L(G').

**Exemple :** Soit le langage de mots composés exclusivement de la lettre 'a'.
Les deux grammaires suivantes sont équivalentes:
G = { {}, {a}, {S -> aS | Sa | a}, S }
G' = { {}, {a}, {S -> aS | a}, S }

**Note** : la relation $\overset{*}{\Rightarrow}$ est la fermeture reflexo-transitive de $\Rightarrow$.

23

## Expressions arithmétiques

- *Preuve d'appartenance*
  - Chemin de dérivation
  - Arbre syntaxique

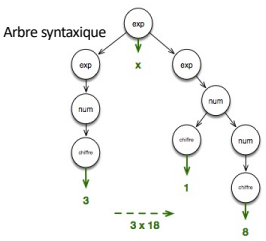
```
exp ::= exp + exp
      | exp × exp
      | (exp)
      | num
num ::= chiffre num
      | chiffre
chiffre ::=  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

L'expression 3 * 18 est-elle une expression arithmétique?

Chemin de dérivation

$exp \Rightarrow exp \times exp \Rightarrow num \times exp$
$\Rightarrow chiffre \times exp \Rightarrow 3 \times exp$
$\Rightarrow 3 \times num \Rightarrow 3 \times chiffre\ num$
$\Rightarrow 3 \times 1\ num \Rightarrow 3 \times 1\ chiffre$
$\Rightarrow 3 \times 18.$

Arbre syntaxique

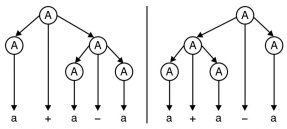



24

## Ambiguïté

**Définition** (Grammaire ambiguë) :
Une grammaire G est dite **ambiguë** si il existe un mot w de L(G) qui admet plusieurs arbres syntaxiques différents.

Exemple : la grammaire G définie par la règle suivante est ambiguë : A → A + A | A - A | a




**Définition** (Langage ambiguë) :
Un langage L est **ambiguë** si toutes les grammaires qui l'engendrent sont ambigues.

Exemple : le langage L(G) n'est pas ambiguë : A → A + a | A – a | a

Extrait wikipedia

25

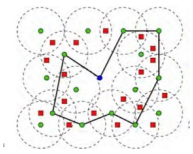
## Problème de l'appartenance

❑ *Quel est le lien entre un problème et la question de l'appartenance?*

➢ Intuitivement tout ce que nous évoquons habituellement sous le terme de problème peut se ramener à une question de décision.

**Définition** (Problème de décision de l'appartenance) :
En théorie des automates, un **problème** se ramène à la question de savoir si un mot donné appartient à tel langage. Plus précisément, soit $\Sigma$ un alphabet, et $L$ un langage sur $\Sigma$, on définit le **problème de décision de l'appartenance** de la façon suivante :

Soit un mot $w$ sur $\Sigma^*$, décider si $w$ est dans $L$ ou non.



https://www.laas.fr/public/fr/sujets-de-recherche-ras

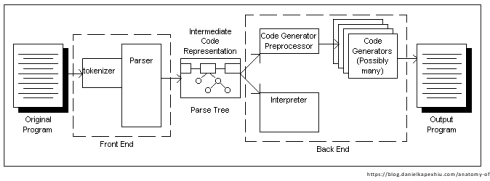
26

## Réduction de problème

❑ ***Réduction d'un problème dans un autre***

➢ En théorie de la compexité on s'intéresse à démontrer les bornes inférieures de la complexité de certains problèmes .

**Définition** (Réduction) :
On appelle **réduction** la technique permettant de montrer qu'un problème est difficile **en supposant** par contradiction l'existence d'un algorithme efficace pour résoudre ce problème **et en appliquant** cet algorithme pour résoudre les instances d'un autre problème déjà connu pour être difficile. Ce qui serait **absurde**.




https://blog.danielkapeshlu.com/anatomy-of-a-compiler

27

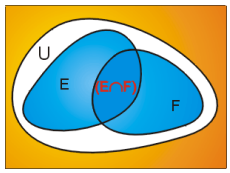
## Opérations ensemblistes sur les langages

**Définition** (opération ensembliste) Union – Intersection :
Soient $L_1$ resp. $L_2$ deux langages sur $\Sigma_1$ resp. sur $\Sigma_2$, alors nous avons :

$$L_1 \cup L_2 = \{w \mid w \in (\Sigma_1 \cup \Sigma_2)^* \; et \; w \in L_1 \; ou \; w \in L_2\}$$
$$L_1 \cap L_2 = \{w \mid w \in (\Sigma_1 \cup \Sigma_2)^* \; et \; w \in L_1 \; et \; w \in L_2\}$$

On définit de façon similaire le **complémentaire** et la **différence** ensembliste :

$$\overline{L_1} = \{w \mid w \in \Sigma_1^* \; et \; w \notin L_1\}$$
$$L_1 \setminus L_2 = \{w \mid w \in \Sigma_1^* \; et \; w \in L_1 \; et \; w \notin L_2\}$$




28

## Opérations de concaténation

**Définition** (opération ensembliste) Concaténation :
Soient $L_1$ resp. $L_2$ deux langages sur $\Sigma_1$ resp. sur $\Sigma_2$, alors nous avons :

$$L_1.L_2 = \{w \mid w \in (\Sigma_1 \cup \Sigma_2)^* \; et \; \exists \; (x, y) \in L_1 \times L_2 \; tel \;\; que \; w = x.y\}$$

Exemples

Soient $L_1 = \{\epsilon, a, ba, \}$ et $L_2 = \{bb, bab\}$, on a :
$L_1.L_2 \;=\; \{bb, bab, a.bb, a.bab, ba.bb, ba.bab\}$

Soient $L_1 = \{\epsilon, a, aa, aaa, ...\}$ et $L_2 = \{b, bb\}$, on a :
$L_1.L_2 \;=\; \{b, bb, ab, abb, aab, aabb, aaab, aaabb, ...\}$

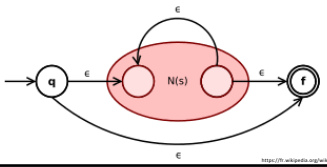
29

## Fermeture de Kleene d'un langage

L'opération de concaténation étant associative on peut définir par récurrence le langage $L^n$ de la façon suivante :
- $L^0 = \{\epsilon\}$ ;
- $L^n = L.L^{n-1} \; \forall n \geq 1$.

**Définition** (Etoile de Kleene) Fermeture de Kleene d'un langage :
Soit un langage $L$, on appelle **fermeture de Kleene** de $L$, notée $L^*$ l'ensemble suivant : $L^* = \bigcup_{k=0}^{\infty} L^k = L^0 \cup L^1 \cup L^2 \cup ...$

$$L^+ = \bigcup_{k=1}^{\infty} L^k$$
$$L^* = L^+ \cup \{\epsilon\}$$




30

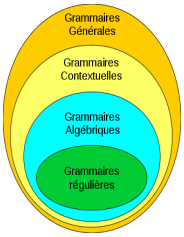
## Hiérarchie de Chomsky

❑ *Quatre classes de grammaires et de langages hiérarchiquement imbriquées*

➢ Chacune des 4 grandes classes de grammaires est définie par des restrictions successives dans la forme des règles.

**Hiérarchie de Chomsky :**

- Les langages de type **0**, les plus généraux sont les langages **récursivement énumérables**;
- Les langages de type **1**, sont les langage**s contextuels**;
- Les langages de type **2** sont appelés les langages **algébriques** ou **hors contextes;**
- Les langages de type **3** sont les langages **réguliers** ou **rationnels**.




https://fr.wikipedia.org/wiki/Hi%C3%A9rarchie_de_Chomsky

31

## Pour résumer

❑ *Quelques points essentiels de l'exposé*

✓ **Définitions des concepts :** alphabet, mot, langage, problème de décision de l'appartenance;

✓ **Les opérations :** union, concaténation, étoile de Kleene d'un alphabet et d'un langage;

✓ **Définition :** Grammaire, dérivation, chemin de dérivation, ambiguité

✓ **Concepts** de problème, problème d'appartenance, réduction de problèmes;

✓ **Hiérarchie de Chomsky :** langage régulier, langage algébrique, langage récursivement énumérable

32

## Bibliographie


- [HMU] **Introduction to Automata Theory, Langage and Computation**
  *J.E. Hopcroft, R. Motwani, J.D. Ullman* Edition Adison Wesley 2001;
- [DPPA10] **Logicomix**
  *A. Doxiadis, C. Papadimitriou, A. Papadatos, A.D. Donna* Edi. Vuibert 2010;
- [Berry 00] **Support de cours de Théorie des Langages**
  *A. Berry – Université Clermont-Auvergne 2000 – 2016*
- [Wikipedia] **Multiples références dans le texte.**


33