Chapitre 1 : Introduction générale

1 Introduction générale

- Les langues et les systèmes de communication sont essentiels pour l'échange d'informations et d'idées entre les personnes,
- Les langages permettent l'interaction entre les humains et les machines.
- Les langues utilisées par les personnes sont généralement ambiguës, nécessitent l'intelligence humaine pour une interprétation adéquate.
- Les langages formels conçus pour la communication avec les ordinateurs sont clairs et dépourvus d'ambiguïté pour garantir une interprétation correcte par les machines.

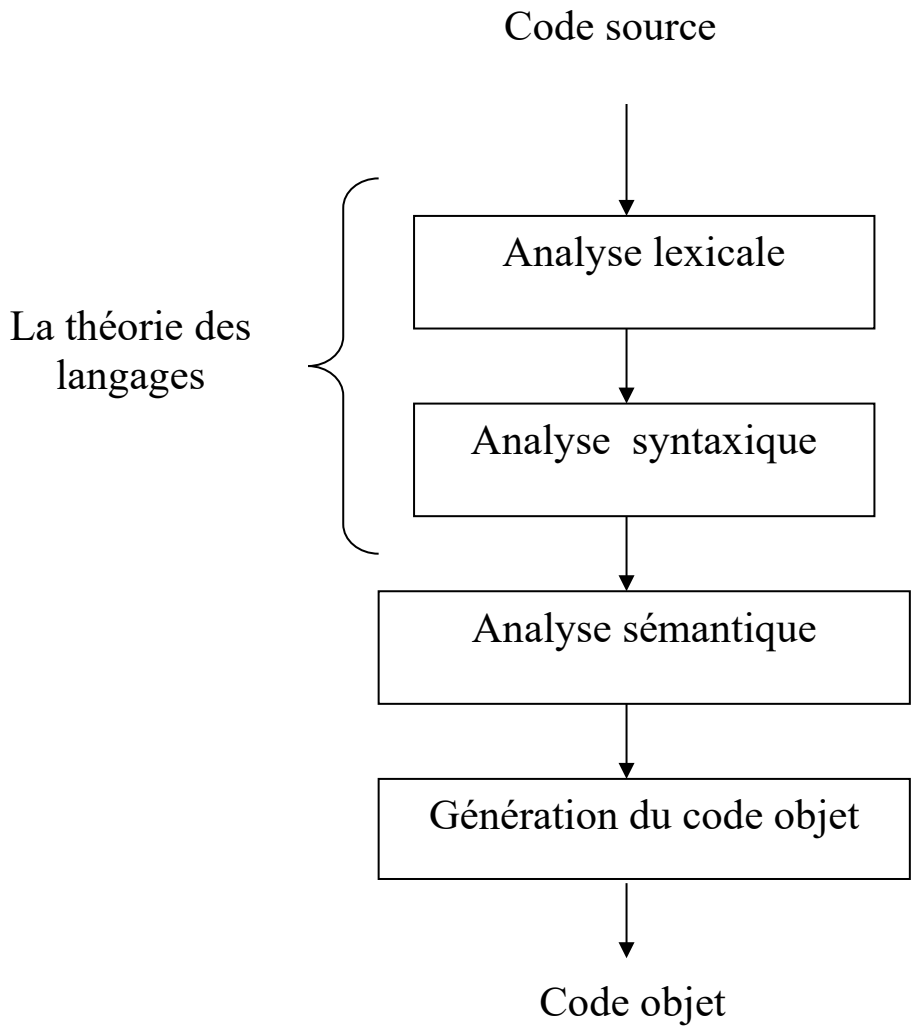
Initialement, un ordinateur ne reconnaît que son propre langage machine, spécifiquement conçu pour lui. Pour interagir avec des langages plus complexes ou de haut- niveaux, on utilisera soit un interprète (qui traduit interactivement les instructions entrées au clavier) soit un compilateur (qui convertit un programme entier).

Le processus de compilation se divise habituellement en trois phases principales :

1. L'analyse lexicale, qui décompose le texte en unités de base nommées lexèmes.

2. L'analyse syntaxique, qui identifie des associations de lexèmes formant des structures syntaxiques.

3. L'analyse sémantique et la génération du code objet, où le code objet est immédiatement compréhensible par la machine, ou sous une forme intermédiaire qui nécessite une conversion ultérieure en code machine.



Les analyses lexicale et syntaxique traitent essentiellement du même défi, bien que sur des niveaux différents.

Dans les deux cas, il s'agit d'identifier une combinaison valide d'éléments : des combinaisons de caractères pour les lexèmes en analyse lexicale, et des combinaisons de lexèmes pour former des programmes en analyse syntaxique.

La théorie des langages fournit les outils pour aborder ce type de problème.

En théorie des langages:

- L'ensemble des entités élémentaires est appelé **l'alphabet**.
- Une combinaison d'entités élémentaires est appelée un **mot**.
- Un ensemble de mots est appelé un langage et est décrit par **une grammaire**.
- A partir d'une grammaire, on peut construire une procédure effective (appelée **automate**) permettant de décider si un mot fait partie du langage.

Il existe différentes classes de langages, correspondant à différentes classes de grammaires et d'automates.

Dans les prochains chapitres nous étudierons la classe des **langages réguliers**, correspondant aux grammaires régulières et aux automates finis. Cette classe de grammaire est typiquement utilisée pour décrire les entités lexicales d'un langage de programmation.

Ainsi nous étudierons la classe des **langages hors contexte**, correspondant aux grammaires hors contexte et aux automates à pile. Cette classe de grammaire, plus puissante que la classe des grammaires régulières, est typiquement utilisée pour décrire la syntaxe d'un langage de programmation.

Chapitre 2 : Introduction aux langages et typologies des grammaires

1. Alphabet, mot et langage

Définition 1 : Alphabet

Un **alphabet** est un ensemble fini non vide des éléments appelés symboles ou lettres. Il est également appelé le vocabulaire. Un alphabet sera en général désigné par une lettre grecque majuscule.

Par ex : $X = \{a, b, c\}, X = \{0, 1\}$, sont des alphabets.

Définition 2 : Mot

Soit X un alphabet. Un **mot** sur X est une suite finie (et ordonnée) de symboles.
Par exemple :

- abbac et ba sont deux mots sur l'alphabet {a, b, c}.
- Le mot 1011 est défini sur l'alphabet {0, 1}

Remarques :

- La longueur d'un mot w est le nombre de symboles constituant ce mot; on la note |w|. par ex : |abbac| = 5 et |ba| = 2 ;
- On définit également la cardinalité d'un mot *w* par rapport à un symbole a ∈ X : $|w|_a$ comme étant le nombre d'occurrence de a dans w. Par exemple, $|abc|_a = 1$, $|aabba|_b = 2$.
- L'unique mot de longueur 0 est le mot correspondant à la suite vide. Ce mot s'appelle le mot vide et on le note ε.
- On note X^+ l'ensemble des mots de longueur supérieure ou égale à 1 que l'on peut construire à partir de l'alphabet X.
- On note X^* l'ensemble des mots que l'on peut construire à partir de X, y compris le mot vide : $X^* = \{\varepsilon\} \cup X^+$

Par exemple, $\{a, b, c\}^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, \ldots\}$

Définition 3 : Concaténation

Soit deux mots x et y de X^*. On définit la concaténation de x et y comme la juxtaposition de x et y. On note xy.

Ex : Si $x = a_1 \ldots a_m$ et $y = b_1 \ldots b_n$ alors $xy = c_1 \ldots c_k$ $[k=m+n]$ avec : $c_i = a_i$ $\forall i \in [1, m]$, et $c_i = b_i$ $\forall i \in [m+1, m+n]$.

Propriété de la concaténation

Soient w, w_1 et w_2 trois mots définis sur l'alphabet X :

- $|w_1 w_2| = |w_1| + |w_2|$;
- $\forall a \in X : |w_1 w_2|_a = |w_1|_a + |w_2|_a$;
- $(w_1 w_2) w_3 = w_1 (w_2 w_3)$ (la concaténation est associative) ;
- $w\varepsilon = \varepsilon w = w$ (ε est un élément neutre pour la concaténation) ;

L'exposant

La concaténation ww est notée par w^2. En généralisant, on note $w^n = \underbrace{w \ldots w}_{n\ fois}$. En particulier, l'exposant 0 fait tomber sur ε : $w^0 = \varepsilon$.

Mot inverse

On appelle mot inverse ou miroir d'un mot w, noté w^R le mot obtenu en inversant les lettres de w. Ainsi si $w = a_1 \ldots a_m$ alors $w^R = a_m \ldots a_1$

- **Ex :** w=ab → w^R=ba.

Préfixe et suffixe

Soit w un mot défini sur un alphabet X.
Un mot x (resp. y) formé sur X ∪ {ε} est un préfixe (resp. suffixe) de w s'il existe un mot u formé sur X∪ {ε} (resp. v formé sur X∪ {ε}) tel que w = xu (resp. w = vy).
Si w = $a_1a_2...a_n$ alors tous les mots de l'ensemble {ε, a_1, a_1a_2, $a_1a_2a_3$, ..., $a_1a_2...a_n$} sont des préfixes de w.
De même, tous les mots de l'ensemble {ε, a_n, $a_{n-1}a_n$, $a_{n-2}a_{n-1}a_n$, ..., $a_1a_2...a_n$} sont des suffixes de w.

Mots conjugués

Deux mots x et y sont dits conjugués s'il existe deux mots u et v tels que : x = uv et y = vu.

Définition 4 : Langage

Un langage sur X est simplement un ensemble (fini ou infini) de mots sur X. En d'autres termes, un langage est une partie de X^*.

Ex :

Considérons l'alphabet X ={a, b, c}.
L'ensemble L1={a, aa, bbc, ccca, ababab} est un langage fini.
L'ensemble L2 des mots sur X comprenant un nombre pair de a est aussi un langage (infini), L2 = {b, c, aa, bb, bc, cb, cc, aab, aac, aba, aca,…abaacaaa}.

Opérations sur les langages :

- Union: $L_1 \cup L_2 = \{w/\ w \in L_1 \text{ ou } w \in L_2\}$. ∪ notée aussi par + ou |
- Intersection: $L_1 \cap L_2 = \{w/\ w \in L_1 \text{ et } w \in L_2\}$,
- Concaténation: $L_1L_2 = \{w_1w_2/\ w_1 \in L_1 \text{ et } w_2 \in L_2\}$,
- Puissance: $L^n = \{w_1 \dots w_n/\ w_i \in L \quad \text{pour tout } i \in \{1, \dots, n\}\}$
- $L^+ = L + L^2 + L^3 + \dots + L^n$.
- Etoile: $L^* = \cup_{n \geq 0} L^n$. Où $L^0 = \{\varepsilon\}$

En particulier, si L = X on obtient X^* c'est-à-dire l'ensemble de tous les mots possibles sur l'alphabet X. On peut ainsi définir un langage comme étant un sous-ensemble quelconque de X^* ;

Propriétés des opérations sur les langages

Soit L, L1, L2, L3 quatre langages définis sur l'alphabet X :

- $L^* = L^+ + \{\varepsilon\}$;
- $L_1.(L_2.L_3) = (L_1.L_2).L_3$;
- $L_1.(L_2 + L_3) = (L_1.L_2) + (L_1.L_3)$;
- $L.L \neq L$;
- $L_1.(L_2 \cap L_3) \neq (L_1 \cap L_2).(L_1 \cap L_3)$;
- $L_1.L_2 \neq L_2.L_1$.
- $(L^*)^* = L^*$;
- $L^*.L^* = L^*$;
- $L_1.(L_2.L_1)^* = (L_1.L_2)^*.L_1$;
- $(L_1 + L_2)^* = (L_1^* L_2^*)^*$;
- $L_1^* + L_2^* \neq (L_1 + L_2)^*$

2. Grammaires (Système générateur de langage)

2.1 Introduction

- Un langage étant défini comme un ensemble de mots, une première façon de décrire un langage est d'énumérer tous les mots qui le constituent.
- Cette méthode se heurte à une difficulté fondamentale où elle ne permet pas de décrire les langages composés d'un nombre infini de mots.
- De plus, même si le langage que l'on désire décrire est fini, ce mode de description ne permet pas de mettre en évidence ce qui est commun dans la structure des mots de ce langages.

- C'est la raison pour laquelle on introduit la notion de grammaire comme méthode de définition d'un langage.
- Une grammaire peut offrir un ensemble de règles fini peut engendrer tous les mots du langage.

Les phrases du langage naturel, comme par exemple : Une phrase simple est constituée d'un sujet suivi d'un verbe suivi d'une préposition suivie d'un complément. Le sujet est un prénom ou bien un pronom. Les prénoms sont à choisir parmi « Mohamed », « Omar » et « Samir ». Les pronoms sont à choisir parmi « Il » et « Elle ». Les prépositions sont à choisir parmi « à », « de », « au » et « en ». Les verbes sont à choisir parmi « va » et « vient ». Les compléments sont à choisir parmi « l'école » et « marché ». Les règles informelles ci-dessus peuvent être décrites par des règles de grammaire.

2.2 Définition formelle d'une grammaire

Une grammaire est un quadruplet G= (T, N, S, R) tel que:

- T est l'alphabet terminal, i.e., l'alphabet sur lequel est défini le langage.
- N est l'alphabet non terminal, i.e., l'ensemble des symboles qui n'apparaissent pas dans les mots générés, mais qui sont utilisés au cours de la génération. Un symbole non terminal désigne une « catégorie syntaxique ».
- R est un ensemble de règles dites de réécriture ou de production de la forme : $u_1 \rightarrow u_2$ avec : $u_1 \in (N \cup T)^+$ et $u_2 \in (N \cup T)^*$. La signification intuitive de ces règles est que la suite non vide de symboles terminaux ou non terminaux u_1 peut être remplacer par la suite éventuellement vide de symboles terminaux ou non terminaux u_2.
- S∈N est le symbole de départ ou axiome. C'est à partir de ce symbole non terminal que l'on peut commencera la génération de mots au moyen des règles de la grammaire.

Ex1:

Soit la grammaire G (T, N, S, R) tel que : T= {0, 1}, N={S}, P= {S→0S1 / 01}

Intuitivement, cette grammaire permet de générer les mots 01, 0011, 000111,….

Le langage défini, ou généré, par une grammaire est l'ensemble des mots qui peuvent être obtenus à partir du symbole de départ par application des règles de la grammaire. Plus formellement, on introduit les notions de *dérivation entre mots*, d'abord en une étape, ensuite en plusieurs étapes:

2.3 Dérivation dans une grammaire

2.3.1 Dérivation directe

Soit une grammaire G=(T, N, S, R), $u \in (N \cup T)^+$ et $v \in (N \cup T)^*$. La grammaire G permet de dériver v de u en une étape (dérivation directe et noté $u \Rightarrow v$) si et seulement si:

- $\boldsymbol{u = xu' y}$ (*u* peut être décomposé en *x*, *u'* et *y*; *x* et *y* peuvent être vides),

- $\boldsymbol{v = xv' y}$ (v peut être décomposé en *x*, *v'* et *y* ; *x* et *y* peuvent être vides),

- $\boldsymbol{u' \rightarrow v'}$ est une règle de R

Ex2 :

Soit G({0, 1},{S}, S, {S→0S1 / 01}),

- 01 dérive directement de S : $S \Rightarrow 01$,

- 00S11 dérive directement de 0S1 : $0S1 \Rightarrow 00S11$

2.3.2 Dérivation indirecte

Une forme *v* peut être dérivée d'une forme *u* en plusieurs étapes:

- $u \Rightarrow^* v$: si *v* peut être obtenue de *u* par une succession de 0, 1 ou plusieurs dérivations,

- $u \Rightarrow^+ v$: si *v* peut être obtenue de *u* par une succession de 1 ou plusieurs dérivations,

Ex3 :

Soit G({0, 1},{S}, S, {S→0S1 / 01}),

- 00S11 dérive de S : $S \Rightarrow^* 00S11$, ou on peut écrire : $S \Rightarrow^2 00S11$

2.4 Le langage engendré par une grammaire

Définition:

Le langage généré par une grammaire *G= (T, N, S, R)* est l'ensemble des mots sur *T* qui peuvent être dérivés à partir de *S*.

$$L(G) = \{v \in T^* \quad / S \Rightarrow^* \quad v\}$$

Ex4 :

Soit G({0, 1},{S}, S, {S→0S1 / 01}), trouver L(G) ?

On a : $L(G) = \{v \in T^* \quad / S \Rightarrow^* \quad v\}$,

Alors : L(G)={$0^n\ 1^n\ /\ n \succ 0$}

Remarques:

1- Une grammaire définit un seul langage. Par contre, un même langage peut être engendré par plusieurs grammaires différentes,
2- On dit que les deux grammaires G1 et G2 sont équivalents si et seulement si L(G1)= L(G2).

2.5 Mot engendré par une grammaire

Soit une grammaire *G* et soit un mot *w*. Pour vérifier si $w \in L(G)$?. Il existe deux manières de répondre à cette question :

1- à travers de dérivations successives à partir de l'axiome, c-à-d en utilisant directement la définition de la dérivation.
2- à travers ***l'arbre de dérivation***. On définit un arbre de dérivation pour un mot donnée, par :
   - le sommet de l'arbre est l'axiome,
   - les branches sont les dérivations successives,
   - les feuilles de l'arbre sont les symboles terminaux.

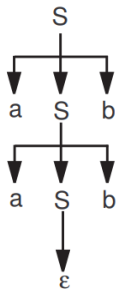
Ex : (un ex plus simple pour bien comprendre…)

Soit la grammaire G({a, b}, {S}, S, P) tel que P comporte les règles :
S → aSb|ε
Le mot aabb est généré par cette grammaire. On peut confirmer ça par deux méthodes :

1- Méthode de dérivation successive :
S => aSb=> aaSbb=> aaεbb = aabb
par conséquent le mot aabb appartient au L(G).

2- Méthode d'arbre de dérivation :

Oui aabb ∈L(G), il existe un arbre de dérivation pour ce mot.



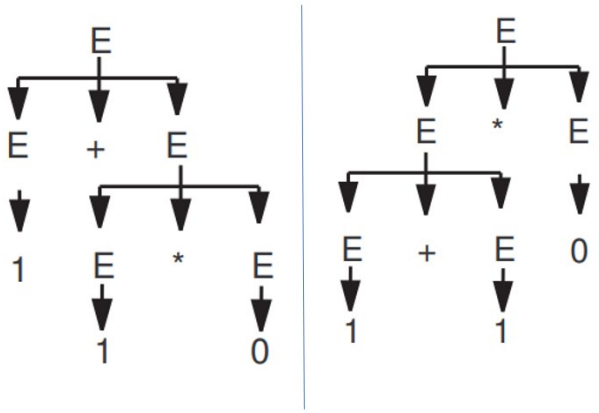
2.6 Grammaire ambiguë

Soit G une grammaire de type *i*.

- Un mot est dit ambigu s'il existe plusieurs arbres de dérivations (ou plusieurs suites de dérivations à partir de S) pour ce mot,
- Une grammaire est **ambiguë**, s'il existe au moins un mot ambigu engendré par G.

Ex :

Soit la grammaire G = ({0, 1, +, ∗}, {E}, E, {E → E + E|E ∗ E|(E)|0|1}). Cette grammaire est ambiguë car le mot 1+1*0 possède deux arbres de dérivation



2.7 Les types de grammaires (Classification de CHOMSKY)

En introduisant des critères plus ou moins restrictifs sur les règles de production, on obtient des classes de grammaires hiérarchisées, ordonnées par inclusion. La classification des grammaires, définie en 1957 par Noam CHOMSKY, distingue les quatre classes suivantes:

- Grammaire de type 0

Toutes les règles sont de la forme :

$$\alpha \rightarrow \beta$$

Tel que : $\alpha \in (N \cup T)^+ - T^+$ et $\beta \in (N \cup T)^*$

Ex :

Par exemple :

aAbb → ba, ou aAbB → ε sont des règles valides, alors que ab → ba ou ε → aA ne les sont pas.

- Grammaire de type 1 (à contexte lié ou sensible au contexte)

Grammaires sensibles au contexte ou contextuelles. Les règles de R sont de la forme: $\alpha \rightarrow \beta$

Avec:

$|\alpha| \leq |\beta|$ et $\alpha \in (N \cup T)^+ - T^+$ et $\beta \in (N \cup T)^*$

Exception : la règle S→ ε est autorisée.

Ex :

La grammaire décrit le langage $\{a^n b^n c^n / n >= 1\}$. Ses règles ont la particularité d'avoir un membre droit de taille supérieure ou égale à celle de leur membre gauche.

$$G = (\{S,A,B,C\},\{a,b,c\},P, S)$$

$$P = \{ S \rightarrow aSBC/abC$$
$$CB \rightarrow BC$$
$$bB \rightarrow bb$$
$$bC \rightarrow bc$$
$$cC \rightarrow cc \ \}$$

- Grammaire de type 2 (algébrique, à contexte libre, hors-contexte ou grammaire de Chomsky)

Les règles de R sont de la forme : $A \rightarrow \beta$

Avec $A \in N$ et $\beta \in (N \cup T)^*$. Autrement dit, le membre de gauche de chaque règle est constitué d'un seul symbole non terminal.

Ex :

La grammaire : $S \rightarrow \varepsilon \ / aSb$. Décrit le langage $\{a^n b^n / n >= 0\}$. Cette grammaire a une particularité : le membre gauche de chaque règle est un non-terminal.

- Grammaire de type 3 (grammaire linaire ou grammaire régulière)

a) Grammaires régulières étendues

- ***Grammaires régulières étendues à droite***. Les règles de R sont de la forme :

$A \rightarrow aB$ ou $A \rightarrow a$ Avec $A, B \in N$ et $a \in T^*$

- ***Grammaires régulières étendues à gauche***. Les règles de R sont de la forme :

$A \to Ba$ ou $A \to a$ Avec $A, B \in N$ et $a \in T^*$

b) Grammaires régulières strictes

- ***Grammaires régulières strictes à droite***. Les règles de R sont de la forme :

$A \to aB$ ou $A \to a$ Avec $A, B \in N$ et $a \in T$

Remarque : la règle S→ ε est autorisée

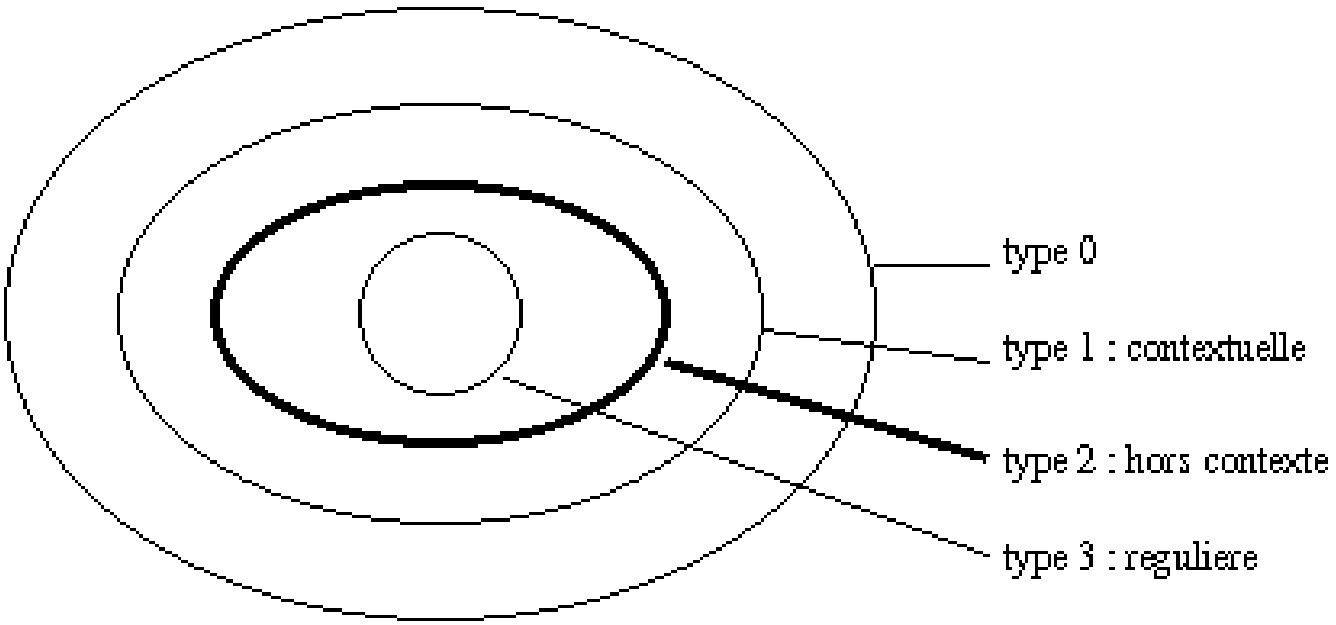
- ***Grammaires régulières strictes à gauche***. Les règles de R sont de la forme :

$A \to Ba$ ou $A \to a$ Avec $A, B \in N$ et $a \in T$

Remarque : la règle S→ ε est autorisée

Ex :…

Propriété : Les grammaires de type 0 englobent les grammaires de type 1 qui englobent les grammaires de type 2 qui englobent les grammaires de type 3.



2.8 Types de langages

A chaque type de grammaire est associé un type de langage: les grammaires de type 3 génèrent les langages réguliers, les grammaires de type 2, les langages hors-contexte et les grammaires de type 1, les langages contextuels. Les grammaires de type 0 permettent de générer tous les langages "décidables" (autrement dit, tous les langages qui peuvent être reconnus en un temps fini par une machine).

Remarque: Les langages qui ne peuvent pas être générés par une grammaire de type 0 sont dits "indécidables".

2.9 Types d'automates (systèmes reconnaisseurs de langages)

A chaque type de grammaire est associé un type d'automate qui permet de reconnaître les langages de sa classe, càd un système qui répond par oui ou non à la question "est-ce que le mot *x* appartient au langage ? Pour les langages réguliers sont reconnus par des automates d'états finis, les langages hors-contexte sont reconnus par des automates à pile, et les autres langages, décrits par des grammaires de type 1 ou 0, sont reconnus par des machines de Turing. Ainsi, la machine de Turing peut être considérée comme le modèle de machine le plus puissant qu'il soit, dans la mesure où tout langage (ou plus généralement, tout problème) qui ne

peut pas être traité par une machine de Turing, ne pourra pas être traité par une autre machine.

Références et sources :

1- Pierre Berlioux, Mnacho Echenim et Michel Lévy, polycope , Théorie des langages ,Année 2017-2018
2- LIF15 – Théorie des langages formels Sylvain Brandel 2015-2016 sylvain.brandel@univ-lyon1.fr
3- Théorie des langages Notes de Cours François Yvon et Akim Demaille avec la participation de Pierre Senellart 19 Janvier 2015