Chapitre 01

Les grammaires:

→ Une grammaire est 4 uplet

$$G = (T, N, S, P)$$

- T : terminaux (a, b, c, 0, 1...)
- N : non terminaux (A, B, C...)
- S : Axiom
- P : Règles de production

Classification des grammaires: 4 types

→ **Type 03 (Grammaire régulière):**
- Régulière droite: ($A \to wB$ ou $A \to w$)
- Régulière gauche: ($A \to Bw$ ou $A \to w$)

→ **Type 02 (Grammaire algébrique):**
Il faut que le membre gauche soit constitué d'un seul non-terminal ($A \to aBb, A \to aBBa$)

→ **Type 01 (Grammaire contextuelles):**
Seul l'axiome peut générer le mot vide

→ **Type 0 (Grammaire général):**
une grammaire est de type 0 si les règles de production dans P n'ont aucune restriction

$$\text{TYPE 03} \subseteq \text{TYPE 02} \subseteq \text{TYPE 01} \subseteq \text{TYPE 0}$$

Hiérarchie de Chomsky

→ c'est quoi le mot vide ?
- l'élément neutre
- mot ne contient pas de lettres

→ quel est le type le plus intéressant ?
- Type 03

Chapitre 02

A chaque type de langage, on associe un type d'automate

- Type 03 — Automates d'état finis
- Type 02 — Automates à piles
-

Automate d'états finis:

→ Un automate EF déterministe est un 5 uplet

$$A = (X, Q, q_0, \nabla, F)$$

- X : alphabet d'entré
- Q : ensbl des états
- q_0 : etat initial
- ∇ : les transitions
- F : etats Finaux

→ $\nabla(P, a) = q \iff$ transition dans de l'état P vers l'état q en liscant la lettre a

→ Un langage peut être reconnu par plusieurs automates. par contre un automate ne peut reconnaître qu'un seul langage

→ un mot est reconnu ssi
- l'automate a terminé la lecture du mot.
- Se trouve dans un état final

les variantes d'AEF:

→ simple et déterministe : Lettres et pas de choix (transition)

→ simple non-déterministe : lettres avec choix

→ partiellement généralisé : Lettre ou ξ

→ généralisé : Mot ou ξ

→ complet : si on peut lire n'importe quel lettre de n'importe quel état

Étapes de transformation :

① Eliminer les transitions par mot en ajoutant des états intermédiaires

② Eliminer les transition spontanées ε

③ Ajouter les états finaux

Automate vers ⇒ Grammaire :

les états = Non terminaux

l'état initial = l'axiom (S)

les lettres lues par transition = Règles de prodn

chaque état final est représenté par une production d'arrêt avec ε

Creation d'automates à partir d'autres automates :

$\overline{L}$: - ajouter un état puit pour rendre (complet)
- inverser les états finaux et non-finaux
- l'état initial ne change pas.

L^r : - inverser les sens des transitions.
- inverser entre l'état final et initial.
- si on a plusieurs états finaux on crée un nouvel état initial qui pointera vers les anciens états finaux avec ε

$L_1 \cup L_2$: - ajouter un nouvel état initial qui pointera avec ε vers les états initiaux de L_1 et L_2 qui deviendront des états normaux.

$L_1 . L_2$: - l'état final de L_1 n'est plus final et pointe vers l'états initial de L_2

→ pour démontrer qu'une grammaire qu'un Langage est regulier on dispose de 3 méthodes :
① une grammaire régulière qui lui génère.
② un automate EF.
③ une expression qui le dénote.

→ le rôle des expressions régulaires : dénoter les langage réguliers.

→ si L_1 et L_2 sont réguliers alors $L_1 \cup L_2$ et $L_1 . L_2$ sont aussi réguliers.

Chapitre 03

Les automates à piles

$$P = (\Sigma, \Gamma, Z_0, Q, q_0, F, \tau)$$

- Σ : Alphabets d'entré
- Γ : Alphabets auxiliaires ($Z_0, Z_1, \ldots$)
- Z_0 : l'état initial
- Q : ensemble des états
- q_0 : état initial
- F : état final
- τ : fonction de transition (u, q, a)
  - u : Sommet pile
  - a : symbole d'entré

MODES de Reconnaissances :

→ par état final
→ par pile vide

dans la configuration final :
→ le contenu de la pile n'est pas important,
→ le mot d'entré a été entièrent lu.
→ l'automate se trouve dans un état final.

→ Automate à pile est un reconnaissant d'un LA
→ tout langage regulier est algébrique
→ Arbre de dérivation :
racine = Axiom
feuilles = terminaux + mot vide
→ mot ambigüe : s'il possède plusieurs arbres de dérivations.
→ Grammaire ambigüe : si elle génère au moins un mot ambigüe.
→ Ambiguité inherente : si toutes grammaire qui l'engendre est ambigüe