

Chapitre 01

Les grammaires :

→ une grammaire est 4 uplet

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

↓
terminaux (a, b, c, 0, 1, ...)

↓
non terminaux (A, B, C, ...)

↓
Axiom

↓
Règles de production

Classification des grammaires : 4 types

→ Type 03 (Grammaire régulière) :

↳ Régulière droite : ($A \rightarrow wB$ ou $A \rightarrow w$)

↳ Régulière gauche : ($A \rightarrow Bw$ ou $A \rightarrow w$)

→ Type 02 (Grammaire algébrique) :

Il faut que le membre gauche soit constitué d'un seul non-terminal ($A \rightarrow aBb$, $A \rightarrow aBbA$)

→ 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

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

↓
alphabet d'entrée

↓
exbi des états

↓
état initial

↓
les transitions

↓
états finaux

→ $\Delta(p, a) = q \Leftrightarrow$ transition de l'état p vers l'état q en lisant 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 ϵ

→ généralisé : Mot ou ϵ

→ 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 transitions spontanées ϵ
- ③ Ajouter les états finaux

Automate \Rightarrow Grammaire :

- les états = non terminaux
- l'état initial = l'axiome (S)
- les lettres lues par transition = Règles de production
- chaque état final est représenté par une production d'arrêt avec ϵ

Création d'automates à partir d'autres automates :

- 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.

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

\rightarrow pour démontrer qu'un langage est régulier on dispose de 3 méthodes :

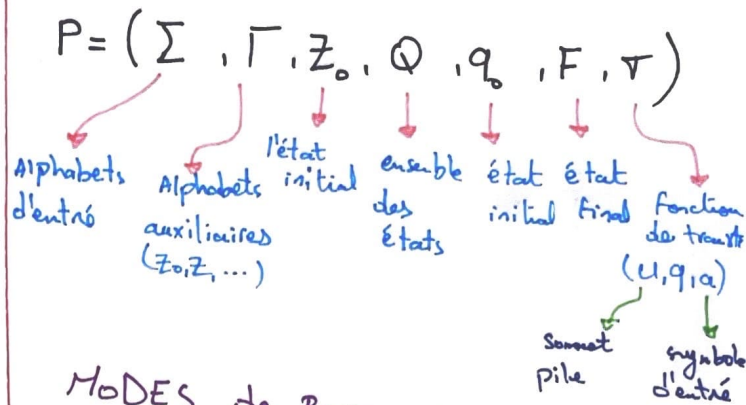
- ① une grammaire régulière qui lui génère.
- ② un automate EF.
- ③ une expression qui le dénote.

\rightarrow le rôle des expressions régulières : dénoter les langages réguliers.

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

chapitre 03

Les automates à piles



MODES de Reconnaissance :

- \rightarrow par état final
- \rightarrow par pile vide
- dans la configuration finale :
- \rightarrow le contenu de la pile n'est pas important.
- \rightarrow le mot d'entrée a été entièrement lu.
- \rightarrow l'automate se trouve dans un état final.

- \rightarrow Automate à pile est un reconnaissant d'un LA
- \rightarrow tout langage régulier est algébrique

Arbre de dérivation :

racine = Axiome
feuilles = terminaux + mot vide

- \rightarrow mot ambiguë : s'il possède plusieurs arbres de dérivation.
- \rightarrow Grammaire ambiguë : si elle génère au moins un mot ambiguë.
- \rightarrow Ambiguïté inhérente : si toutes les grammaires qui l'engendrent sont ambiguës