THÉORIES DES LANGAGES

Mr,HEMIOUD
hemourad@yahoo,fr
Université de Jijel
Département d'informatique

Langages hors-contexte (algébriques) et Automates à pile

Certains langages ne peuvent pas être décrits par une ***grammaire régulière***, et ***ne peuvent*** donc pas être ***reconnus par un automate fini*** (par exemple le langage $\{a^n b^n / n > 0\}$).

On étudie dans ce chapitre une classe de langages plus générale que celle des langages réguliers : la classe des ***langages hors-contexte***, décrits par des grammaires hors-contexte et reconnus par des ***automates à pile***.

Grammaire hors-contexte :

G = (T, N, S, *R*) est une grammaire hors-contexte si toutes les règles de *R* sont de la forme $\mathbf{A} \rightarrow \boldsymbol{w}$ avec $A \in N$ et $w \in (N \cup T)^*$.

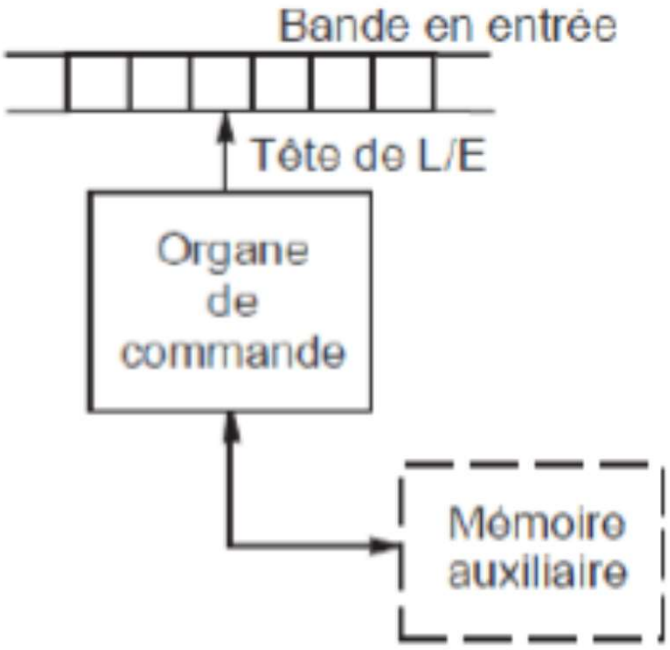
- ***Langage hors-contexte*** : On appelle langage hors-contexte un langage généré par une grammaire hors contexte.

Les automates a piles

Les automates à états fini

- Un **automate** est une machine abstraite qui permet de lire un mot et de répondre à la question : "**un mot *w* appartient-il à un langage *L* ?**" par **oui** ou **non**.

- Un automate est composé de :



Limite des automates finis

- Certains langages ne peuvent pas être reconnus par les automates finis (ne peuvent être générés par une grammaire régulière)

Exemple : $L = \{a^n b^n \mid n \geq 0\}$

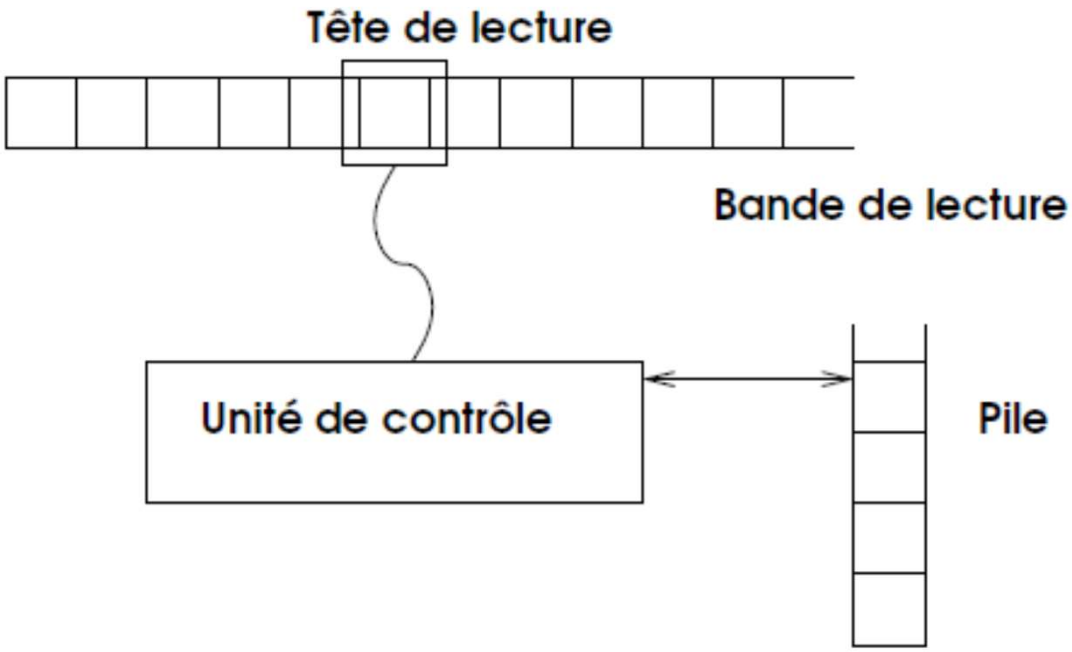
- Il faut ***mémoriser*** le nombre de ***a*** que l'on a lu pour vérifier que le mot possède autant de ***b***.
- Pour mémoriser un nombre potentiellement infini de ***a***, il faut un ensemble ***infini*** d'états !

AUTOMATES À PILE

Principe : Les ***AuP*** fonctionnent sur le même principe que les AEF : depuis un état ***p***, ils consomment un caractère du mot et effectue la transition correspondante qui les amène dans un nouvel état ***q***

À la différence des AEF, à chaque transition ils mettent à jour une pile et peuvent ainsi enregistre des informations utiles pour la reconnaissance.

AUTOMATES À PILE



À la différence des AEF, à chaque transition ils mettent à jour une pile et peuvent ainsi enregistre des informations utiles pour la reconnaissance

Généralités

- Forme simple de ***mémoire*** : une **pile**.
  - Mode de stockage **Last In First Out.**
  - on accède à la pile **uniquement** par son sommet
- Deux ***opérations*** possibles :
  - **empiler** : ajouter un élément au sommet.
  - **dépiler** : enlever l'élément se trouvant au sommet.
- La pile permet de stocker de l'information ***sans*** forcément ***multiplier*** le *nombre d'états*.

Configurations

- **Configuration initiale**
  - L'unité de contrôle est dans un **état initial**
  - La tète est au **début** de la bande
  - La mémoire contient un élément initial.(**la pile est vide)**
- **Configuration d'acceptation**
  - L'unité de contrôle est dans un **état d'acceptation**
  - La tète de lecture est à **la fin** de la bande
  - La mémoire se trouve dans un état d'acceptation (***la pile est vide***)

Un automate à pile non-déterministe (APN) est un septuple (Q,A,P,δ,q_0,Z,Q_F) avec :

- **Q** : ensemble fini d'états
- **A** : alphabet fini des symboles d'entrée
- **P** : alphabet fini des symboles de pile (a priori $P \cap A = \emptyset$)
- q_0 : état initial
- $Z \in P$: symbole initial de pile
- $Q_F \subseteq Q$: ensemble des états terminaux
- **δ** est l'ensemble des règles de transition

- une règle $\delta(\boldsymbol{p}, \boldsymbol{a}, \alpha) = (\mathbf{q},\beta)$ de transition considère :
  - l'état courant ***p*** de l'automate
  - le caractère lu ***a*** sur le ruban (ou peut-être pas : **ε**)
  - le symbole **α** de **sommet** de pile (ou peut-être pas : Z)
- une **règle** indique :
  - le prochain état ***q*** de l'automate
  - la suite de symboles **β** à **empiler** à la place du **sommet** de pile

Configurations et mouvement

- **Configuration** : $(q, w, \alpha) \in Q \times A^* \times P^*$ où :
  - ***q*** représente l'état courant de l'unité de contrôle
  - ***w*** est la partie du mot à reconnaitre non encore lue. Le premier symbole de w (le plus à gauche) est celui qui se trouve sous la tète de lecture. Si $\mathbf{w = \varepsilon}$ alors tout le mot a été lu.
  - α représente le contenu de la pile. Le symbole le plus à gauche est le sommet de la pile. Si $\alpha = \mathbf{Z}$ alors la pile est ***vide***.

- Configuration **initiale** : (q_0, w, Z) où w est le mot à reconnaitre

- Configuration **d'acceptation** : (q, ε, Z) avec $q \in Q_F$

- **Mouvement** :
  - $(p, \boldsymbol{a}w, B) \vdash (q, w, AB)$ (si $\delta(p, a, B) = (q, AB)$)
  - $(p, \boldsymbol{a}w, AB) \vdash (q, w, B)$ (si $\delta(p, a, A) = (q, \varepsilon)$)
  - $(p, \boldsymbol{a}w, A) \vdash (q, w, A)$ (si $\delta(p, a, B) = (q, A)$)

- **Représentation graphique**

Exemple 1

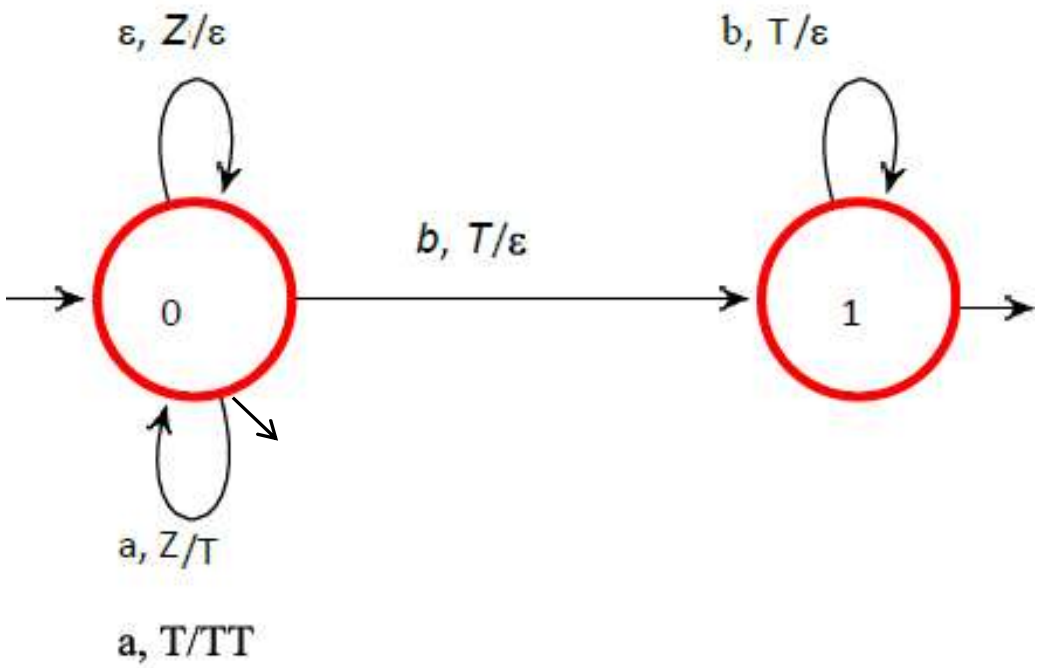
- Soit l'automate à pile suivant qui reconnaît le langage $\{a^n b^n / n \geq 0\}$

↘

- **Représentation graphique**

Exemple 1

- Soit l'automate à pile suivant qui reconnaît le langage $\{a^n b^n / n \geq 0\}$



- **Exemple** 2
- Soit l'automate à pile suivant qui reconnaît le langage $\{w \in \{a, b\}^* \mid w$ est un *palindrome*$\}$

Simplification des grammaires hors-contextes

Simplification des grammaires hors-contextes

1. **Les grammaires propres**

   Une grammaire hors-contexte (V, N, S, R) est dite **propre** si elle vérifie :

   - $\forall A \rightarrow u \in R : u \neq \varepsilon$ ou $A = S$;
   - $\forall A \rightarrow u \in R$: S ne figure pas dans u ;
   - $\forall A \rightarrow u \in R : u \notin N$;
   - Tous les non terminaux sont ***utiles***, c'est-à-dire qu'ils vérifient :
     - $\forall A \in N$: A est <u>atteignable</u> depuis S : $\exists \alpha, \beta \in (N + V)^*$: $S \rightarrow^* \alpha A \beta$;
     - $\forall A \in N$: A est <u>productif</u> : $\exists w \in V^*$: $A \rightarrow^* w$.

- Il est toujours possible de trouver une grammaire propre pour toute grammaire hors contexte. En effet, on procède comme suit :

1. **Rajouter** une nouvelle règle **S′ → S** tel que **S′** est le nouvel **axiome ;**
2. **Éliminer** les règles **A → ε** :
   - Calculer l'ensemble E = {A ∈ N ∪ {S′} | A→*ε } ;
   - Pour tout A ∈ E, pour toute règle B → αAβ de R
   - Rajouter la règle B →αβ
   - Enlever les règles A → ε ;
3. **Eliminer** les règles **A→* B**, on applique la procédure suivante sur R privée de S′→ ε :
   - Calculer toutes les paires (A, B) tel que A→* B
   - Pour chaque paire (A, B) trouvée
     - Pour chaque règle B → u_1 | … | u_n rajouter la règle A → u_1 | … | u_n
   - Enlever toutes les règles A → B
4. **Supprimer** tous les non-terminaux **non-productifs**
5. **Supprimer** tous les non-terminaux **non-atteignables**