

THÉORIE DES LANGAGES ET DES AUTOMATES

CH4: AUTOMATES À PILES





PLAN

- Généralités.
- Définition.
- Exemple introductif
- Transitions dans un PDA
- Configurations
- Langage reconnu par un PDA

AUTOMATE À PILE



- L'approche pour *analyser la syntaxe du code source d'un programme* est de scanner le programme *de gauche à droite*, en cherchant une dérivation qui génère ce programme.
- Le modèle de base d'un programme qui fait une telle analyse est un *automate à pile*.
- C'est une généralisation de la notion *d'automate fini* à des grammaires hors-contexte.
- Comme expliqué auparavant, les automates finis peuvent analyser seulement la syntaxe d'une grammaire régulière.
- Pour analyser la syntaxe d'une grammaire hors-contexte, nous ajoutons une pile à un automate fini pour obtenir un modèle de programmes plus puissant connu sous le nom de "*automate à pile*".

LIMITE DES AUTOMATES FINIS



- Nous avons vu qu'un automate fini est un modèle de programme avec un nombre fini d'états.
- Par conséquent, il a nombre fini d'actions ou transitions qu'il peut effectuer.
- Comme il a un nombre fini d'états et de transitions, il est très simple à programmer et même à visualiser graphiquement.

AUTOMATE À PILE

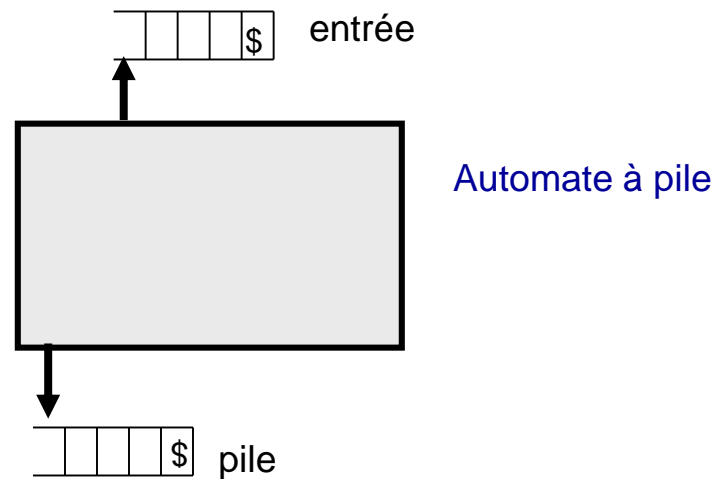


- Un automate à pile ressemble à un automate fini. Il a :
 - Une mémoire de lecture et un pointeur vers le prochain symbole à lire;
 - un nombre fini d'états, y compris un état initial et des états accepteurs;
 - une relation de transition.
- La différence est qu'un automate à pile a en plus une pile, initialement vide, mais pouvant croître arbitrairement.
- Le contenu de la pile fait partie de l'état d'un automate.
- Donc un automate à pile a potentiellement un nombre infini d'états.



AUTOMATE À PILE

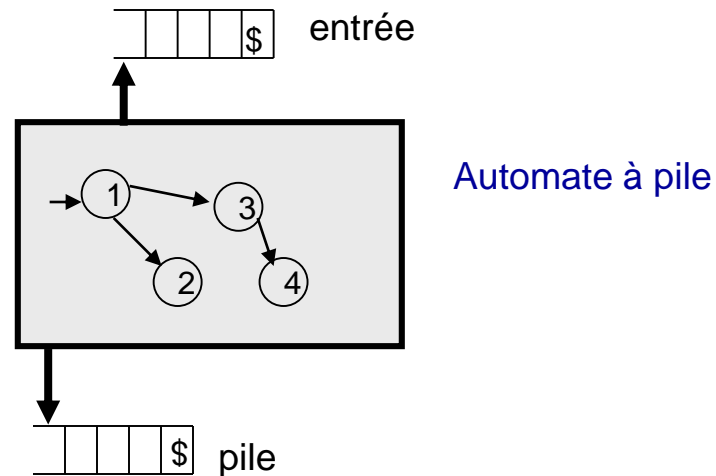
- Un automate à pile a :
 - Une entrée et une tête de lecture.
 - Un nombre fini d'états dont un état initial et des états accepteurs.
 - Une relation de transition.
 - Une pile pouvant croître arbitrairement.



AUTOMATE À PILE



- L'entrée est une chaîne de *tokens* scanné à partir du code source du programme à analyser.
- La pile contient une chaîne de symbole de la grammaire (terminaux et non-terminaux)
- Les transitions indiquent comment mettre à jour le contenu de la pile en fonction du *token* courant.



AUTOMATE À PILE : DÉFINITION FORMELLE



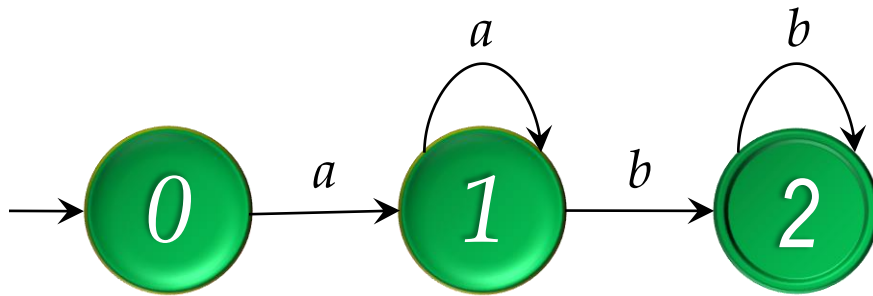
Un automate à piles $\mathcal{A} = \langle E, \Sigma, \Pi, \delta, e_0, z_0, F \rangle$

- E : un ensemble fini d'états.
- Σ un ensemble de symboles (l'alphabet des symboles d'entrée).
- Π est un alphabet fini dit l'alphabet de la pile.
- $\delta: \Pi^* \times E \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(\Pi^* \times E)$, une fonction finie.
- Un état $e_0 \in E$: état de départ ou état initial.
- z_0 est le symbole initial de la pile.
- Un ensemble d'états $F \subseteq E$ connus comme états d'acceptation



EXEMPLE INTRODUCTIF

considérons l'automate fini \mathcal{A} de la Figure suivante, qui reconnaît le langage $\{a^n b^m, \text{ avec } n > 0 \text{ et } m > 0\}$ et essayons d'étendre ce modèle pour en dériver un automate à pile capable de reconnaître $\{a^n b^m, m = n > 0\}$.

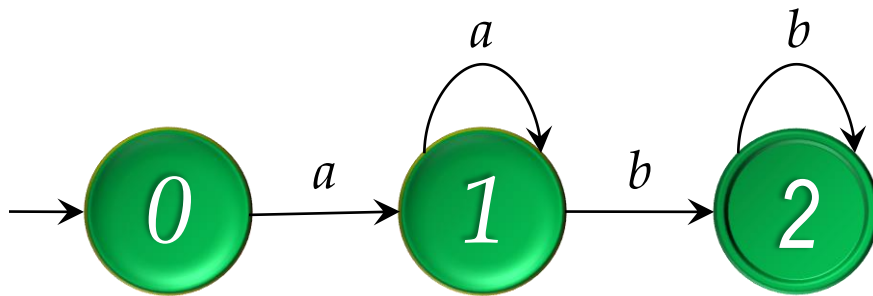


L'automate \mathcal{A} est incapable de “compter” le nombre de a déjà vus, ce qui interdit d'imposer la contrainte $n = m$.

Confions alors cette tâche au mécanisme de pile, en empilant un symbole Z lors de chaque passage dans la boucle de l'état 1 et en dépilant un symbole lors de chaque passage dans la boucle de l'état 2. Si l'on impose, de plus, qu'un calcul réussi dans \mathcal{A} doit correspondre à une pile intégralement vide, alors il apparaît que chaque mot reconnu par l'automate fini \mathcal{A} augmenté de la pile est bien tel que le nombre de a est exactement égal au nombre de b .



EXEMPLE INTRODUCTIF

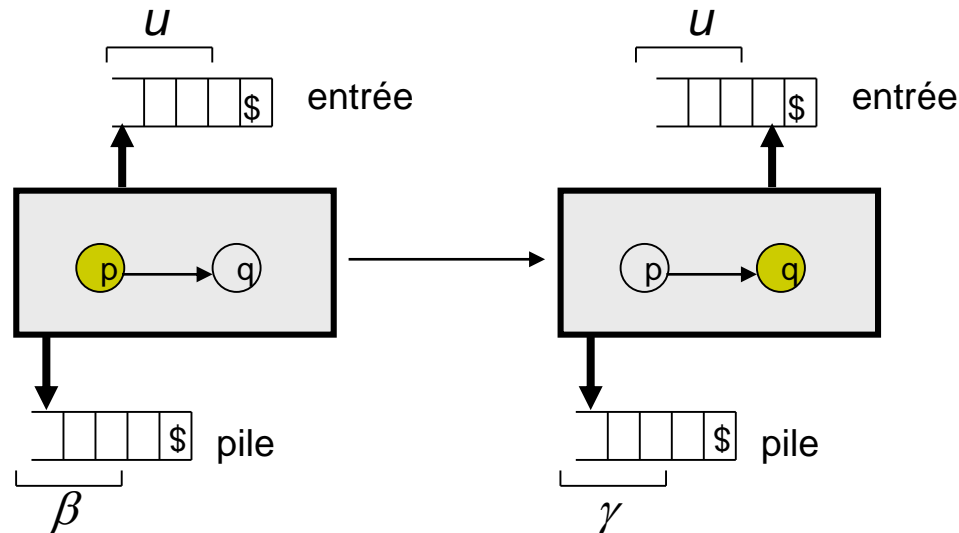


<i>input</i>	<i>state</i>	<i>stack</i>
<i>aaabbb</i>	0	ε
<i>aabbb</i>	1	ε
<i>abbb</i>	1	Z
<i>bb</i>	2	Z Z
<i>b</i>	2	Z
	2	ε



TRANSITIONS

- Une *transition* de la forme $(p, u, \beta) \rightarrow (q, \gamma)$ signifie que l'automate peut passer de l'état p à l'état q , pourvu que la chaîne d'entrée commence par le préfixe u et la chaîne β est au sommet de la pile.
 - Après la transition, u est lu et consommé du mot d'entrée, β est enlevé de la pile et remplacé par γ , et le nouvel état devient q .
 - La chaîne β est lue de gauche à droite : son premier caractère doit donc être au sommet de la pile.
 - La chaîne γ est écrite de telle façon que son premier caractère soit au sommet de la pile.

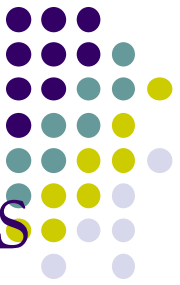




CONFIGURATIONS

- La *configuration* d'un automate est défini comme suit: (u, q, x, γ)
u: mot lu,
q: état courant,
x: le reste à lire,
 γ : contenu de la pile.
- *L'exécution d'un automate sur une entrée* est la séquence de configurations qu'il produit en lisant son entrée et en suivant ses transitions.

.



AUTOMATES À PILE ET AUTOMATES TRADITIONNELS

De manière sous-jacente, un automate à pile est essentiellement un automate fini non-déterministe, à la différence près que la fonction de transition comporte trois arguments : l'état courant, le symbole d'entrée courant et le symbole courant en haut de la pile.

Si (Z, e) est un élément de (Y, q, a) , alors l'utilisation de cette transition conduira à :

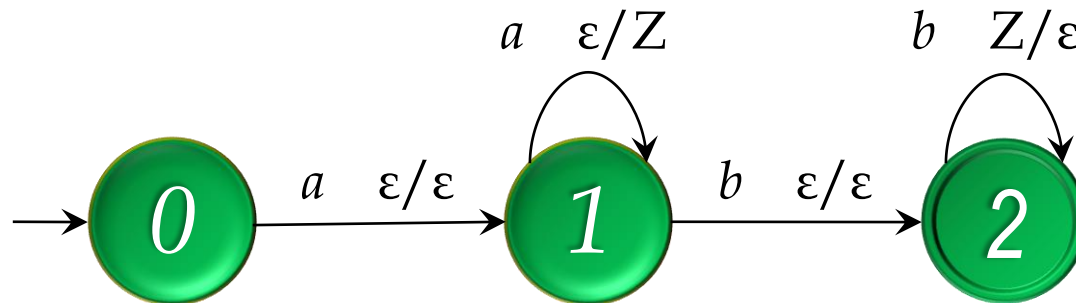
- Dépiler Y ; si $Y = \varepsilon$ la transition a lieu indépendamment du symbole en haut de pile, qui reste inchangée.
- Transiter dans l'état r .
- Empiler Z ; si $Z = \varepsilon$, aucun symbole n'est empilé.

AUTOMATES À PILE ET AUTOMATES TRADITIONNELS



Compte-tenu de cette définition, un automate fini “traditionnel” est un automate à pile particulier, défini sur un alphabet de pile vide ($\Pi = \emptyset$) et dont toutes les transitions laissent la pile inchangée.

Un automate à pile admet une représentation graphique sur laquelle les modifications de la pile sont représentées sous la forme Y/Z . La Figure suivante représente ainsi un automate à pile reconnaissant le langage $\{a^n b^n\}$.





LANGAGE RECONNU PAR UN PDA

- Le langage $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$ reconnu par \mathcal{A} , par **état final** est défini par : $u \in \mathcal{L}(\mathcal{A})$ si et seulement si il existe $\pi \in \Pi^*$ et $s \in F$ tels que :

$$(\varepsilon, q_0, w, Z_0) \quad \vdash_{\mathcal{A}}^* \quad (w, q_f, \varepsilon, Z_0)$$

- Le langage $\mathcal{L}_\varepsilon(\mathcal{A}) \subseteq \Sigma^*$ reconnu par \mathcal{A} , par **pile vide** est défini par : $u \in \mathcal{L}_\varepsilon(\mathcal{A})$ si et seulement si il existe $q \in E$:

$$(\varepsilon, q_0, w, Z_0) \quad \vdash_{\mathcal{A}}^* \quad (w, q, \varepsilon, \varepsilon)$$

- Pour tout PDA \mathcal{A} , il existe un PDA \mathcal{A}' tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}_\varepsilon(\mathcal{A}')$

AU DELÀ DES LANGAGES HORS-CONTEXTE



- Bien qu'un automate à pile peut avoir un nombre infini d'états (configurations), il existe des langages qui ne sont pas acceptés par un automate à pile, c-à-d., des langages qui ne sont pas hors-contextes.
- Exemple:

$$L(M) = \{a^n b^n c^n \mid n \geq 0\}$$

n'est pas hors-contexte.

- Toutefois, il existe des modèles théoriques de programmes qui accepte un tel langage. En particulier, les *machines de Turing* sont des généralisations des automates à piles, pouvant accepter n'importe quel langage.