7. Automates à Piles (AP)

Bernard ESPINASSE

Professeur à l'Université d'Aix-Marseille

1998

- **Introduction**
- **Définition d'un automate à Pile**
- **Langage reconnu par un automate à pile**
- **Automates à pile déterministes et non déterministes**
- **Passage d'une grammaire hors contexte à un automate à pile**
- **Limites de l'automate à pile, introduction à la Machine de Turing**

Automates à pile : introduction

Limites de l'**A**utomate à **E**tats **F**ini car **faible pouvoir de mémorisation** (réduit à son ensemble d'états)

=> ne permet pas la reconnaissance de langages tels que $\{a^n b^n\}$

Pour augmenter la puissance de l'AEF on peut lui adjoindre un dispositif de **mémorisation** appelé **pile**

-> modèle de l'Automate à Pile (AP)

- utilisé pour :
  - la **reconnaissance** et **l'analyse** des **langages de programmation**
  - dans de nombreux **algorithmes** **(récursifs)**
- fournit la description d'un **mécanisme de calcul** :
  - qui intègre un **ensemble infini d'états**
  - avec **une règle de changement d'état** pouvant être décrite de manière finie

***∃** des **langages** simples qui **ne peuvent être reconnus par AP** : L = {**chaînes** $a^n b^n c^n$}*

Automates à pile : introduction

- associés aux **grammaires hors contexte**
- **plus puissants** que les automates à états finis (reconnaissent langages hors contexte)
- possèdent une **mémorisation des étapes antérieures**
- mémorisation faite au moyen d'une **pile** : les dernières informations stockées sont les premières extraites

Exemple classique :

- langage **hors contexte non régulier** = **{chaînes $a^n b^n$}**
- chaînes **ne peuvent être reconnues par un AEF**, car il ne peut compter le nb de **a** lus lors du parcours de la chaîne $\mathbf{a^n}$
- dans un **automate à pile** :
  - il est possible de stocker une information à chaque **a** lu
  - information utilisée lors du parcours de la chaîne $\mathbf{b^n}$ pour lire le même nombre de **b**

Définition d'un automate à pile

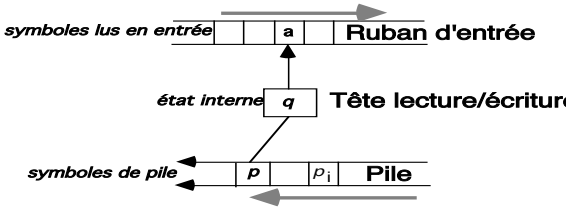
Un automate à pile = 7-uplet **$A = (V, Q, P, q_i, F, p_i, \mu)$** avec :

- **V** = **vocabulaire** = **alphabet d'entrée** = un ensemble fini, non vide,
- **Q** = **ensemble d'états**, fini, non vide,
- **P** = **alphabet de pile**, non vide,
- **q_i** = **état initial** = un état particulier de Q ,
- **F** = les **états finals** = un sous-ensemble d'états particuliers ,
- **p_i** = **symbole de pile initial** = symbole particulier de P
- **μ** = **fonction de transition de l'automate** =
  relation de **$Q \times (V \cup \{\lambda\}) \times P$** -> **$Q \times P^*$** (ex : $\mu(q, a, p) = (q', p')$)

- l'état global d'un AP a 2 composantes :
  - un **élément de l'ensemble fini Q** (l'ensemble des états proprement dits),
  - le **contenu de la pile**
- **μ** donne les modifications possibles de cet état global vers un autre état global
- le **contenu de la pile** permet de décrire **μ** de manière **finie**, en ne considérant que la dernière lettre du mot écrit sur la pile

Fonctionnement intuitif d'un automate à pile

- À un instant déterminé, le **fonctionnement de l'AP est défini par**:
  - le **symbole lu** sur le ruban d'entrée,
  - l'**état** de l'automate,
  - le **symbole** situé sur le **dessus de la pile**.
- Selon les valeurs de ces 3 paramètres l'automate à pile va
  - **changer d'état**
  - **effacer le symbole situé sur le dessus de la pile** et **empiler un mot** (si le mot écrit est le mot vide, ceci revient à dépiler)
  - après quoi, le **ruban d'entrée est avancé d'une case**



- L'AP peut aussi **changer d'état** et **écrire sur la pile sans lire le caractère du ruban d'entrée** (ruban d'entrée immobile (ε-opération)) : manipuler la pile sans avancer le ruban d'entrée.

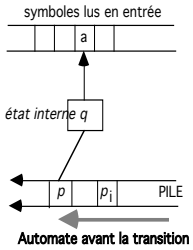
Définition d'un automate à pile

A = (V, Q, P, q_i, F, p_i, μ) alors **μ(q, a, p) = (q', p_n p_{n-1}...p_1)** indique que :

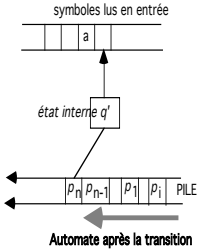
- **si** A est dans l'état **q**, qu'il rencontre le symbole **a**, et qu'il a **p** en sommet de pile:
- **alors** il passe dans l'état **q'**, dépile **p** et empile **p_n p_{n-1}...p_1**

- on peut **vider la pile par des transitions** du type : **μ (q, ∧, p) = (q', ∧)**
- si le symbole lu **=** symbole vide **∧**, alors A n'avance pas sur la chaîne d'entrée (ε-opération)

Exemple : Etat de A avant et après la transition **μ(q, a, p) = (q', p_n p_{n-1}...p_1)** :



Automate avant la transition



Automate après la transition

Exemple 1 d'automate à pile

Soit l'AP **= (V, Q, P, q_i, F, p_i, μ)** reconnaissant le langage L = {$a^n b^n$ / n ∈ N}

- V = {a, b} ;
- Q = {e_0, e_1, e_2 };
- P = {c};
- q_i = e_0 ;
- F = ∅ ;
- p_i = c
- μ est définie ainsi :

  μ(e_0, c, a) = {(e_1, c)}

  μ(e_1, c, a) = {(e_1, cc)}

  μ(e_1, c, b) = {(e_2, ∧)}

  μ(e_2, c, b) = {(e_2, ∧)}

Fonctionnement de l'automate pour le mot **aaabb** :

symbole lu :	↓ **a**aabbb	a↓**a**abbb	aa↓**a**bbb	aaa↓**b**bb	aaab↓**b**b	aaabb↓**b**	↓ aaabbb
mot sur pile :	c	c	cc	ccc	cc	c	∧
état de l'AP :	e_0	e_1	e_1	e_1	e_2	e_2	e_2

Exemple 2 d'automate à pile

Soit l'AP = **(V, Q, P, q_i, F, p_i, μ)** défini par :

- V = {a, b}
- Q = {q_i, q_1, q_2}
- P = {p_i, A }
- q_i = état initial
- F = états finals
- p_i = symbole pile initial

- μ fonction de transition (p = symbole de pile quelconque) :

  μ(q_i, a, p_i) = (q_1, A)

  μ(q_i, a, p) = (q_1, Ap)

  μ(q_i, b, A) = (q_2, ∧)

  μ(q_2, b, A) = (q_2, ∧)

- Dérivations conduisant à **l'acceptation** de la chaîne **aabb** :

  (q_i, aabb, p_i) |- (q_1, abb, A) |- (q_1, bb, AA) |- (q_2, b, A) |- (q_2,∧ , ∧)

  pas de non déterminisme car toutes les transitions de l'automate sont déterministes
- Dérivations conduisant à l'**échec de l'acceptation** de la chaîne **aaba** :

  (q_i, aaba, p_i) |- (q_1, aba, A) |- (q_1, ba, AA) |- (q_2, a, A)

  dans l'état q_2, avec une pile égale à A et le symbole a en entrée, aucune des quatre transitions de l'automate n'est applicable. Il y a donc échec de l'acceptation.

Automates à pile non déterministes

Un AP est **non déterministe** :

Si la fonction de transition μ a pour image, non pas un couple de Q x P*, mais un **ensemble de couples**

=> l'ensemble d'arrivée de μ est alors l'ensemble des parties de Q x P*, noté *P* (Q x P*).

Inconvénients :

- la reconnaissance d'une chaîne **nécessite des retours en arrière** dans le cas **où** le choix fait lors d'une transition non déterministe est incorrect
- **la complexité de l'algorithme de reconnaissance est exponentielle** : alors qu'elle est **linéaire** dans le cas d'un automate déterministe.

Remarque :

- contrairement aux AEF, le **non déterminisme des AP** apporte **une puissance de reconnaissance supplémentaire**

Langage reconnu par un AP

C'est la construction de **{chaînes} acceptées** à l'aide de **dérivations l-** définies à partir de μ. et enchaînant les **transitions correctes** sur un automate donné :

$$\forall q, q' \in Q, \forall a \in V, \forall x \in V^*, \forall p \in P, \forall \pi, \pi' \in P^*,$$
$$(q, ax, p\pi) \vdash (q', x, \pi'\pi) \Leftrightarrow (q', \pi') \in \mu(q, a, p).$$

- **langage accepté par un AP par état final** = {chaînes} reconnues entièrement par un AP et le faisant s'arrêter dans un **état final** :

$$FIN(A) = \{x \in V^* \text{ tels que } (q_i, x, p_i) \vdash^* (q, \wedge, P), q \in F\}$$

- **langage accepté par un AP par pile vide** = {chaînes} reconnues entièrement par un AP et le faisant s'arrêter avec une **pile vide** :

$$VIDE(A) = \{x \in V^* \text{ tels que } (q_i, x, p_i) \vdash^* (q, \wedge, \wedge)\}$$

Théorèmes

- Les classes de langages acceptés par **pile vide** ou par **état final** sont **identiques**
- Tout langage accepté par un automate à pile est **hors contexte**
- **Pour tout langage hors contexte, il ∃ un automate à pile qui l'accepte**

Exemple de langage reconnu par un AP

Soit un AP qui reconnaît **par pile vide** les chaînes miroir composées d'un nombre pair de lettres **a** et **b** telles que **aabaabaa** ou **abbbbbba**

- Il est composé des 5 transitions suivantes :

(1) $\mu(q_i, \wedge, p_i) = (q_i, \wedge)$

suppression initiale du fond de pile,

(2) $\forall p \in P, \mu(q_i, a, p) = (q_i, Ap)$ et $\mu(q_i, b, p) = (q_i, Bp)$

réalise l'empilement des symboles en début de chaîne,

(3) $\mu(q_i, a, A) = (q_2, \wedge)$ et $\mu(q_i, b, B) = (q_2, \wedge)$

(dépilement et changement d'état en milieu supposé de chaîne,

(4) $\mu(q_2, a, A) = (q2, \wedge)$ et $\mu(q_2, b, B) = (q_2, \wedge)$

parcours de la supposée deuxième moitié de chaîne et

(5) $\mu(q_2, \wedge, p_i) = (q_2, \wedge)$

arrivée avec succès en fin de chaîne

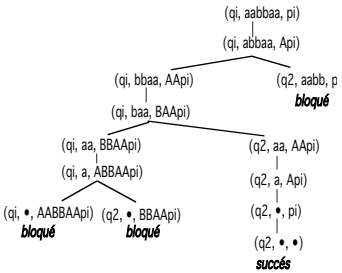
- automate **non déterministe**, car les transitions de milieu de chaîne (3) peuvent être remplacées par des transitions d'avancée dans la première moitié (2).

Exemple de langage reconnu par un AP (suite)

- **Trace** du **fonctionnement** de l'AP sur la chaîne **aabbaa**, sans tenir compte des retours en arrière dans le cas de mauvais choix pour les transitions non déterministes :

$$(q_i, aabbaa, p_i) \vdash (q_i, abbaa, Ap_i) \vdash (q_i, bbaa, AAp_i)$$
$$\vdash (q_i, baa, BAAp_i) \vdash (q_2, aa, AAp_i) \vdash (q_2, a, Ap_i) \vdash (q_2, \wedge, p_i) \vdash (q_2, \wedge, \wedge)$$

- **Arbre d'acceptation de l'AP** avec impasses résultant des mauvais choix dans le cas de transitions non déterministes :



*Analyse de la chaîne **aabbaa** par arrêt sur pile vide* **(• = ∧)**

- On peut réécrire l'AP pour reconnaissance des chaînes miroir par **état final** au lieu de **par pile vide**

Passage d'une Grammaire Hors Contexte à un Automate à Pile (GHC -> AP)

On considère le passage simple acceptant de créer des transitions qui n'avancent pas sur la chaîne d'entrée (transitions avec un symbole lu vide) :

- à **chaque règle non terminale** → une **transition** qui **empile** les **symboles de la partie droite**
- les **symboles terminaux de partie droite** ou les **règles terminales** → des **transitions** qui **dépilent** ces symboles et reconnaissent le caractère attendu

Donc à une règle de la forme **A → BcD**, on associe les **transitions** :

- **$\mu(q_1, \wedge, A) = (q_1, BcD)$ et**
- **$\mu(q_1, c, c) = (q_1, \wedge)$**

- on amorce l'analyse en empilant un symbole S (l'axiome) à l'état initial par :

$$\mu(q_i, \wedge, p_i) = (q_1, S)$$

- l'automate reconnaît alors les chaînes de la grammaire et seulement celles-ci par arrêt sur **pile vide**

Exemple de passage GHC -> AP

Soit A automate à pile qui reconnaît, par **pile vide**, les chaînes de la **GHC** suivante :

- S → aSb
- S → aCb
- C → cC
- C → c

A possède :

- **4 transitions associées aux terminaux de la grammaire :**
  - $\mu(q_i, \wedge, p_i) = (q_1, S)$
  - $\mu(q_1, \wedge, S) = (q_1, aSb)$
  - $\mu(q_1, \wedge, S) = (q_1, aCb)$
  - $\mu(q_1, \wedge, C) = (q_1, cC)$
- **4 transitions de reconnaissance des symboles terminaux :**
  - $\mu(q_1, a, a) = (q_1, \wedge)$
  - $\mu(q_1, b, b) = (q_1, \wedge)$
  - $\mu(q_1, c, C) = (q_1, \wedge)$
  - $\mu(q_1, c, c) = (q_1, \wedge)$

Trace du fonctionnement de l'automate sur la chaîne **aaccbb**, sans tenir compte des retours en arrière dans le cas de mauvais choix pour les transitions non déterministes:

$(q_i, aaccbb, pi)$ I- $(q_1, aaccbb, S)$ I- $(q_1, aaccbb, aSb)$
I- $(q_1, accbb, Sb)$ I- $(q_1, accbb, aCbb)$ I- $(q_1, ccbb, Cbb)$
I- $(q_1, ccbb, cCbb)$ I- (q_1, cbb, Cbb) I- (q_1, bb, bb)
I- (q_1, b, b) I- $(q, \wedge, \wedge)$

Conclusion sur les AP

Le modèle de l'Automate à Pile dépasse les limites de l'automate à Etat Fini :

- AP utilisé pour :
  - la **reconnaissance** et **l'analyse** des **langages de programmation**
  - dans de nombreux **algorithmes** **(récursifs)**
- AP fournit la description d'un **mécanisme de calcul** :
  - qui intègre un **ensemble infini d'états**
  - avec **une règle de changement d'état** pouvant être décrite de manière finie

Le modèle de l'Automate à Pile ayant lui aussi ses limites :

- ∃ des langages simples qui ne peuvent être reconnus par AP : $L = \{a^n b^n c^n\}$
- on peut en augmenter la puissance en lui adjoignant une **deuxième pile** afin de reconnaître ce langage $L = \{a^n b^n c^n\}$
- mais modèle mal aisé à manipuler :

*=> On lui préfère un modèle équivalent : la **Machine de Turing**, qui s'avère être le modèle de calcul le plus général que l'on puisse proposer.*

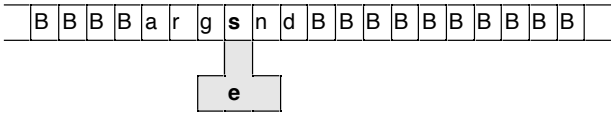
Description intuitive d'une Machine de Turing (MdT)

Une **Machine de Turing M** est composée :

- d'une **mémoire** appelée **bande (ou ruban)** composée d'une suite infinie de cases où sont inscrits des symboles (un symbole par case),
- d'une **tête de lecture**, caractérisée par un nombre fini **d'états**, qui peut **lire** un symbole écrit sur une case de la bande, et **réagir** ainsi :
  - **écrire** un autre symbole sur la case,
  - **changer d'état**,
  - **se déplacer** d'une case vers la **droite** ou vers la **gauche**.
- un ensemble fini de **règles** (le programme de la machine) précisant les "réactions" de la tête de lecture.

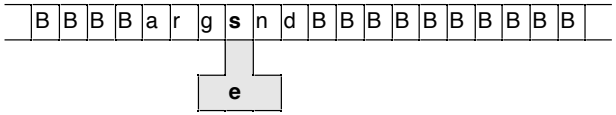
Initialement, seul un nombre fini de cases sont marquées avec un symbole autre que le **symbole spécial B** (**symbole blanc**).

La machine **s'arrête** quand une situation n'est pas prévue par les règles.
Représentation d'une Machine de Turing :



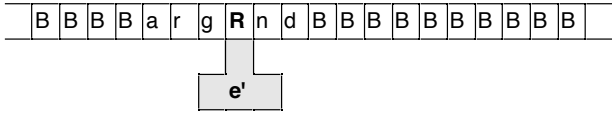
Fonctionnement intuitif d'une Machine de Turing

• Soit une Machine de Turing dans l'état suivant :



Machine de Turing dans un état donné

• Si une **règle** indique que lorsque la machine, étant dans l'état **e**, lit le symbole **s**, elle écrit le symbole **R**, et elle se déplace vers la **droite** et passe dans l'état **e'**, une application de cette règle donnera :



Machine de Turing après application de la règle.

L'**état global de la machine** est constitué de deux parties:

- une partie finie qui est **l'état de la machine** proprement dit,
- une partie, comportant un nombre non borné de **possibilités**, donnée par la suite de symboles écrits sur la bande.

Les **règles** indiquent comment modifier l'état global de la machine.