Chapitre 5 : Automates à pile

Introduction

Dans ce chapitre, nous allons présenter un nouveau type de modèle de calcul qui étend le modèle des automates finis. Il s'agit des **automates à pile** (**Pushdown Automata**).

Un **automate à pile** (**AP** en abrégé) est un automate fini non déterministe équipé d'une **pile** jouant le rôle d'une mémoire de taille non prédéterminée. La pile est une structure qui fonctionne selon la règle « **LIFO** » (**Last In**/**First Out**). Le pointeur de la pile examine toujours l'élément au sommet de la pile (**top**) et ne peut effectuer que deux opérations de base sur la pile : l'empilement d'un élément (**push**) et le dépilement de l'élément au sommet de la pile (**pop**). Voici le schéma général qui illustre le concept d'un AP :

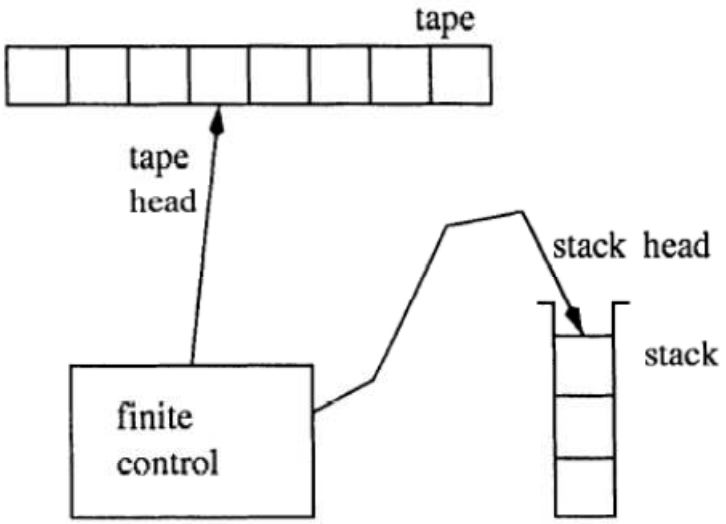

Figure 1. *Schéma général d'un automate à pile*.

Un AP peut reconnaître certains langages non réguliers (comme par exemple les langages $\{a^n b^n \mid n \geq 0\}$ et $\{ww^R \mid w \in \{a, b\}^*\}$). Nous allons voir que les AP sont équivalents aux GHC : tout langage hors-contexte est accepté par un AP, et inversement tout langage accepté

par un AP est hors-contexte. Cela montre que la puissance de calcul des AP est plus grande que celle des AFD ou des AFND, puisque la classe des langages réguliers est incluse dans la classe des langages hors-contexte.

Exemple introductif

Prenons le langage $L = \{a^n b^n \mid n \geq 0\}$. Ce langage étant non régulier, il est donc impossible de l'accepter par un automate fini (déterministe ou non). Cependant, en utilisant une pile d'une manière convenable, on peut reconnaître le langage L.

L'idée est simple et consiste à utiliser la pile pour vérifier qu'il y a exactement n occurrences de b juste après les n occurrences de a. Il suffit alors d'utiliser un automate M avec deux états q_0 et q_1. A l'état q_0, M empile tout simplement chaque symbole a qu'il vient de lire. Après la lecture de tous les symboles a, M passe spontanément à l'état q_1 de manière non-déterministe. A l'état q_1, M dépile chaque symbole a lorsqu'il lit un nouveau symbole b. A la fin de la lecture la pile sera vide s'il y avait autant de b que de a.

Prenons par exemple le mot $w = a^2b^2$.

- Au départ, l'automate se trouve à l'état initial q_0 et la pile est vide.
- Il lit le 1er symbole a et l'empile, et reste dans l'état q_0.
- Il lit le 2eme symbole a et l'empile, et reste dans l'état q_0.
- Maintenant que tous les symboles a ont été lu, il passe spontanément à l'état q_1 sans rien lire.
- Il lit le 1er symbole b et dépile la pile tout en restant dans l'état q_1.
- Il lit le 2eme symbole b et dépile la pile tout en restant dans l'état q_1.
- Maintenant que le mot w est entièrement lu et la pile est vidée, il peut annoncer qu'il a accepté le mot $w = a^2b^2$.

Comme les AF, les AP peuvent être déterministes (APD) ou non-déterministes (APND). Malheureusement, les APD et les APND **ne sont pas équivalents** (ils n'ont pas donc la même puissance de calcul). Cependant, les APD et APND reconnaissent la même classe

de langages : les langages hors-contexte. Nous n'étudierons ici que les APND, car ils sont très intéressants en pratique.

Définition formelle d'un APND

Soit X un alphabet. On pose : $X_\varepsilon = X \cup \{\varepsilon\}$.

Définition 1. Un **APND** est un 6-uplet $M = (E, \Sigma, \Gamma, \delta, q_0, F)$ où :

(1) E : un ensemble fini d'états.

(2) Σ : l'alphabet de l'entrée.

(3) Γ : l'alphabet de la pile.

(4) δ : la fonction de transition définie de $E \times \Sigma_\varepsilon \times \Gamma_\varepsilon$ vers l'ensemble des parties de $E \times \Gamma_\varepsilon^*$: $P(E \times \Gamma_\varepsilon^*)$.

(5) $q_0 \in E$: l'état initial.

(6) $F \subseteq E$: l'ensemble des états finaux.

Voici comment on définit une transition dans un AP. Pour un état $q \in E$, un symbole $a \in \Sigma_\varepsilon$ et un symbole $x \in \Gamma_\varepsilon$, l'ensemble des transitions possibles est noté $\delta(q, a, x)$ qui peut être vide. $\delta(q, a, x)$ est composé d'un nombre fini de couples de la forme (p_i, y_i) où $p_i \in E$ et $y_i \in \Gamma_\varepsilon$: $\delta(q, a, x) = \{(p_1, y_1), (p_2, y_2), \ldots, (p_k, y_k)\}$. Cela signifie que si l'AP M se trouve dans l'état q et si le prochain symbole de l'entrée à lire est a, alors l'un des transitions (on dit aussi mouvements) possibles de M est (p_i, y_i). Une fois le mouvement (p_i, y_i) est pris, l'automate M effectue les actions suivantes : il lit le symbole d'entrée a et avance la tête de lecture vers le symbole suivant (celui à droite de a), dépile le symbole x et empile le symbole y_i, puis passe à l'état p_i. On note cette transition comme suit :

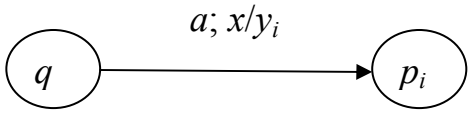

Figure 2. *Schéma d'une transition dans un AP.*

Il est possible que x et/ou y_i soient égaux à ε.

- Lorsque $y_i = \varepsilon$: l'automate lit a et dépile x.
- Lorsque $x = \varepsilon$: l'automate lit a et empile y_i.
- Lorsque $y_i = \varepsilon$ et $x = \varepsilon$: l'automate lit a sans modifier la pile.

Il est également possible d'avoir $a = \varepsilon$. Dans ce cas, l'automate change spontanément son état sans rien lire, mais selon les cas des symboles x et y_i, il peut effectuer ou non des opérations sur la pile.

Au départ, un AP $M = (E, \Sigma, \Gamma, \delta, q_0, F)$ se trouve dans l'état q_0, la pile est vide et sa tête de lecture pointe vers le premier symbole de l'entrée (c'est-à-dire le symbole le plus à gauche). A la fin, nous disons que l'AP M accepte l'entrée, s'il existe un chemin de calcul ξ dans M vérifiant les trois conditions suivantes :

- ✓ ξ garantie que tous les symboles d'entrée sont lus, un après l'autre.
- ✓ ξ garantie que la plie est vide.
- ✓ ξ garantie qu'un état final est atteint.

Déterminons formellement l'AP qui accepte le langage étudié dans l'exemple introductif : $\{a^n b^n \mid n \geq 0\}$.

Exemple 1. $M = (E, \Sigma, \Gamma, \delta, q_0, F)$ où :

- $E = \{q_0, q_1\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{a\}$
- $F = \{q_1\}$
- δ est définie par sa table :

Input	*a*			*b*			*ε*		
Stack	*a*	*b*	*ε*	*a*	*b*	*ε*	*a*	*b*	*ε*
q_0			$\{(q_0, a)\}$						$\{(q_1, \varepsilon)\}$
q_1				$\{(q_1, \varepsilon)\}$					

Table 1. *Table de transition de l'AP du langage* $\{a^n b^n \mid n \geq 0\}$.

Terminologie des AP

Définition 2. Une **configuration** d'un AP M est un élément de l'ensemble $(E, \Sigma_\varepsilon^*, \Gamma_\varepsilon^*)$: $C = (q, w, \gamma)$.

Une configuration permet de mémoriser les informations sur M à une certaine étape du calcul, y compris son état courant q, le mot qui reste à lire w, et le contenu de la pile γ (les symboles empilés du plus haut au plus bas sont écrits de gauche à droite).

Définition 3. On dit que l'AP M **passe en une seule étape** de la configuration $C = (q, w, \gamma)$ à la configuration $C' = (p, w', \gamma')$, et l'on note $C \mapsto C'$, si et seulement si il existe un mouvement $(p, y) \in \delta(q, a, x)$ tel que : $w = aw'$, $\gamma = x\alpha$, et $\gamma = y\alpha$.

Nous pouvons donc écrire : $(q, aw', x\alpha) \mapsto (p, w', y\alpha)$, lorsque le mouvement $(p, y) \in \delta(q, a, x)$ est possible.

Définition 4. On dit que l'AP M **passe en plusieurs étapes** de la configuration $C = (q, w, \gamma)$ à la configuration $C' = (p, w', \gamma')$, et l'on note $C \mapsto^* C'$, si et seulement si il existe $(n + 1)$ configurations C_0, C_1, …, C_n telles que :

- $C_0 = (q, w, \gamma)$;
- $C_i \mapsto C_{i+1}$, pour tout $i = 0, 1, \ldots, n - 1$;
- $C_n = (p, w', \gamma')$.

Définition 5. Soit x un mot de Σ^*. La chaîne de configurations dans le passage en plusieurs étapes de la **configuration initiale** $C = (q_0, x, \varepsilon)$ à une configuration finale $C' = (p, x', \gamma)$ est appelé un **chemin de calcul dans** M **pour le mot** x. La configuration $C' = (p, x', \gamma)$ est dite **finale** s'il est impossible de passer de cette configuration à une autre configuration de M.

Définition 6. Un mot x de Σ^* est **accepté** par l'AP M, s'il existe un chemin de calcul dans M pour x menant vers une configuration de la forme $(q_f, \varepsilon, \varepsilon)$, où q_f est un état final de M ($q_f \in F$). Cela signifie donc que l'on a : $(q_0, x, \varepsilon) \mapsto (q_f, \varepsilon, \varepsilon)$.

Définition 7. Le **langage accepté** par un AP M, noté $L(M)$, est le langage de tous les mots x de Σ^* acceptés par M. Formellement :

$$L(M) = \{x \in \Sigma^* \mid (\exists q_f \in F) : (q_0, x, \varepsilon) \mapsto (q_f, \varepsilon, \varepsilon)\}$$

Intuitivement, le mot x est accepté par l'AP M, si M réussit à partir de son état initial q_0 et partant d'une pile vide, à lire (consommer) tous les symboles de x tout en se retrouvant dans l'un de ces états finaux avec une pile vide.

Exemple 2. Considérons l'AP M suivant représenté par un diagramme de transition :

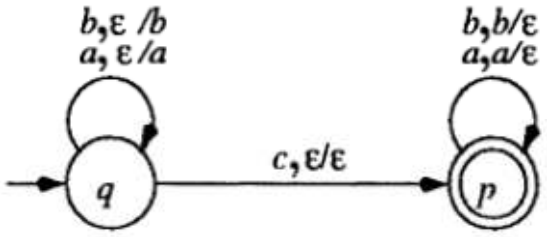

Figure 3. *Un AP pour le langage* $\{wcw^R \mid w \in \{a, b\}^*\}$.

Nous avons :

- $(q, c, \varepsilon) \mapsto (p, \varepsilon, \varepsilon) : c \in L(M)$.
- $(q, aca, \varepsilon) \mapsto (q, ca, a) \mapsto (p, a, a) \mapsto (p, \varepsilon, \varepsilon) : aca \in L(M)$.
- $(q, abcba, \varepsilon) \mapsto (q, bcba, a) \mapsto (q, cba, ba) \mapsto (p, ba, ba) \mapsto (p, a, a) \mapsto (p, \varepsilon, \varepsilon) : abcba \in L(M)$.
- $(q, acb, \varepsilon) \mapsto (q, cb, a) \mapsto (p, b, a) : acb \notin L(M)$.

Il n'est pas difficile de voir que : $L(M) = \{wcw^R \mid w \in \{a, b\}^*\}$.

Remarque importante. Dans un AFD ou un AFND, un chemin de l'état initial vers un état final garanti que le mot est accepté. Ce n'est pas le cas pour un AP, car il faut tenir compte de la pile : c'est **une pile vide avec un état final** qui détermine si un mot est accepté ou non.

Exercice corrigé 1.

Construire un AP pour les langages suivants :

1. $\{a^i b^j c^k \mid i + k = j\}$.
2. $\{a^i b^j \mid 2i = 3j\}$.
3. $\{ww^R \mid w \in \{a, b\}^*\}$.

Solution de l'exercice corrigé 1.

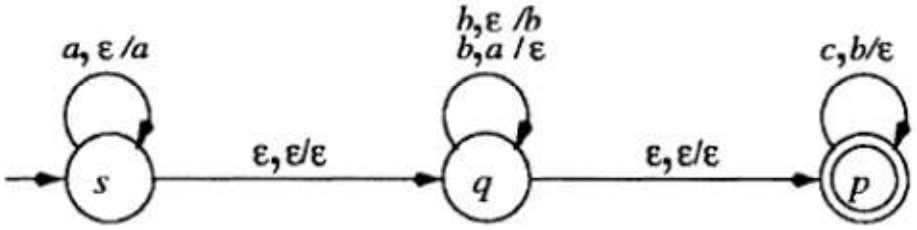

Figure 4. *Un AP pour le langage* $\{a^i b^j c^k \mid i + k = j\}$.

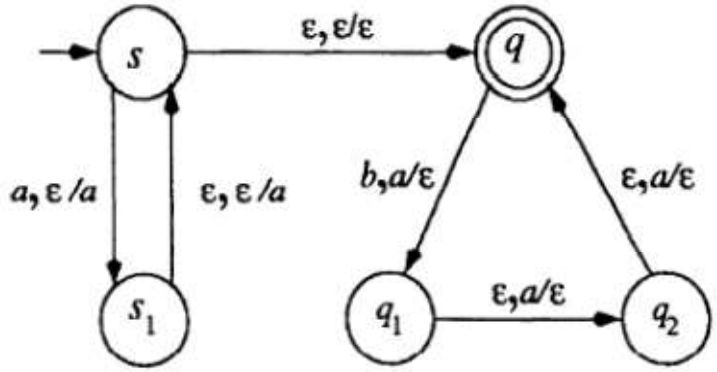

Figure 5. *Un AP pour le langage* $\{a^i b^j \mid 2i = 3j\}$.

.

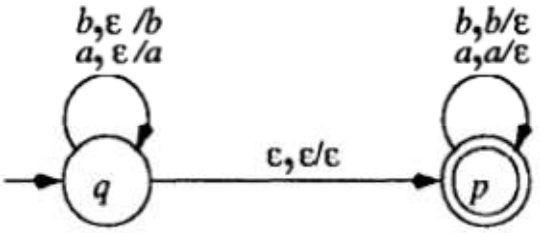

Figure 6. *Un AP pour le langage* $\{ww^R \mid w \in \{a, b\}^*\}$.

Automates à pile & grammaires hors-contexte

Théorème. Un langage est hors-contexte si et seulement il est accepté par un AP.