UNIVERSITÉ TOULOUSE III PAUL SABATIER

Rédigé avec LaTeX
Version du 3 décembre 2013

Langages et automates

L3 Informatique
Semestre 5

Cours donné par Christine Maurel – JP Arcangeli
Rédigé par Antoine de Roquemaurel

2013

Avant-Propos

– 3 séances CM
– 8 séances JPA
– 4 séances CM

0.1 *

MCC

Contrôle continue 30%

Contrôle terminal 70%

0.2 *

Objectifs Avoir les bases pour aborder la compilation lors du S7 en master.

Le but est de nous apprendre ce qu'est un **automate** et ce qu'est une **grammaire**.

Table des matières

0.1 * 2

0.2 * 2

1 Langage **4**

1.1 Introduction 4

1.2 Langages et grammaires 4

1.3 Opérations sur les langages 5

1.4 Grammaire 5

2 Automates à pile **7**

2.1 Introduction 7

2.2 Définition 7

2.3 Configuration 8

2.4 Déterminisme 10

2.5 Automate à pile déduit d'une grammaire 10

3 Machine de Turing **11**

3.1 Informellement 11

3.2 Définition 11

3.3 Configuration 12

3.4 Relation entre configuration 12

3.5 Langages acceptés ; langages décidés 12

3.6 Introduction à la calculabilité 12

Chapitre 2

Automates à pile

2.1 Introduction

Quelque soit le langage et la classe du langage, celui-ci est reconnu par un automate est il est engendré par une grammaire.

Régulier S'exprime avec une expression régulière et engendre un automate fini déterministe et est associé à une grammaire linéaire à droite.
Langage régulier $\Longleftrightarrow$ Expression régulière $\Longleftrightarrow$ Automate fini déterministe $\Longleftrightarrow$ système d'équation de langage $\Longleftrightarrow$ Grammaire linéaire

Langage hors contexte Ce sont les langages ayant besoin d'une mémoire, ils sont reconnus par un automate à pile et engendré par une grammaire hors contexte.

> Par exemple le langage $a^n b^n n \geq 0$ est un langage hors contexte. Il nécessite de connaître le nombre de a pour pouvoir faire les b. Sa grammaire hors contexte serait :
>
> $$\begin{aligned} S &\rightarrow aSb \\ S &\rightarrow \lambda \end{aligned}$$

2.2 Définition

2.2.1 Informelle

Un automate à pile possède le même fonctionnement d'un automate fini avec en plus une pile permettant de sauvegarder de l'information.

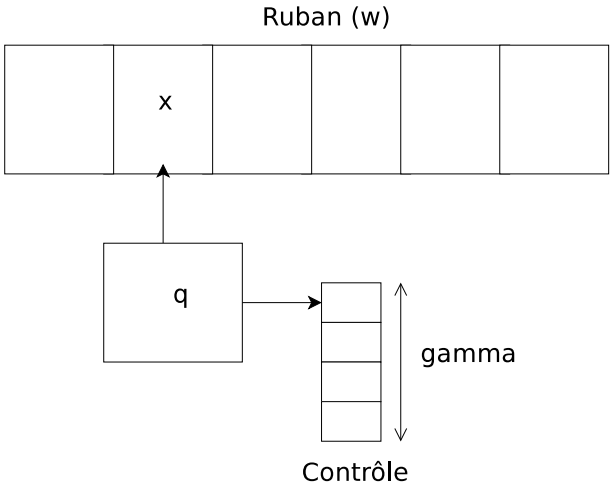

Figure 2.1 – Schéma d'un automate à pile

R Contrairement à un automate finis, un automate à pile peut lire λ sur le ruban.

2.2.2 Formelle

Un automate à pile est un 7-uplet $< Q, X, F, \Gamma, q_0, Z_0, \delta >$
- Q un ensemble finis d'états
- X l'alphabet, un ensemble finis de symboles avec $\lambda \notin X$
- F un ensemble finis d'états finals avec $F \subset Q$, éventuellement $F = \varnothing$
- Γ un ensemble fini de symboles de pile, éventuellement $\Gamma \cap X = \varnothing$
- $q_0 \in Q$, état initial
- $Z_0 \in Gamma$ symbole de fond de pile
- $\delta : Q \times (X \cup \{\lambda\}) \times \Gamma$ [1] $\rightarrow P(Q \times \Gamma^*$ [2]$)$

- On peut accéder uniquement au sommet de pile
- On remplace le sommet de pile Z par un mot de λ^*
- Une pile qui contient Z_0 n'est pas vide

2.3 Configuration

2.3.1 Définition

Une configuration est une « photo de la machine » à un instant donné, celle-ci est un triplet : L'état, le ruban et la pile, (q, w, γ)
- $q \in Q$

1. Un seule symbole $\in \Gamma$
2. Un mot à la plae du sommet de pile

– $w \in X^*$
– $\gamma \in \Gamma$
Les lettres sont positionnées sur la figure 2.1.

2.3.2 Relation entre deux configuration

Soit deux configurations $(q, xu, Z\gamma)$ et $(q', u, \beta\gamma)$

– $q, q \in Q$
– $x \in X \cup \{\lambda\}$
– $Z \in \Gamma, \gamma \in \Gamma^*\beta \in \Gamma^*$

Definition 2.1 La relation $\vdash$ est la dérivation en une étape de calcul (δ si $\delta(q, x, Z)$ contient $(q, xu, Zj) \vdash (q', u\beta\gamma)$

Definition 2.2 La relation $\vdash^*$ exprime l'enchainement des étape de calcul.

2.3.3 Reconnaissance

Triplet (q_0, w, Z_0). Langage reconnu par un automate à pile A deux modes de reconnaissance.
Par état final $T(A) = \{w \in X^*/(q_0, w, Z_0) \vdash^* (qf, \lambda, \gamma)$ avec qf_inF
Par pile vide $N(A) = \{w \in X^*/(q_0, w, Z_0) \vdash^* (q, \lambda, \lambda), q \in Q$
Deux modes de reconnaissance équivalents sont automatiquement à pile non déterministe.

Soit le langage $L = a^n b^n n \geq 0$. Construire un automate à pile A qui reconnait L

Reconnaissance par pile vide.

$A =< Q, X, F, \Gamma, q_0, Z_0, \delta >$

– $Q = \{q_0,$
– $X = \{a, b\}$
– $F = \varnothing$
– $\Gamma = \{Z_0, A$

```
tantque on lit 'a' faire empiler 'A' fin tantque;
tantque on lit 'b' avec 'A' au sommet faire depiler fin tantque;
Le mot est reconnu, si quand on a fini de lire on retrouver Z0 dans la pile
```

– $\delta : Q \times X \cup \{\lambda\} \times \Gamma \to P(Q, \Gamma^*)$

– $\delta(q_0, a, Z_0) = (q_0, AZ_0)$ on empile le premier 'a' lu
– $\delta(q_0, a, A) = (q_0, AA)$ On empile tous les 'a' lus
– $\delta(q_0, b, A) = (q_1, \lambda)$ On dépile A au 1er'b' lu (dépiler = empiler λ)
– $\delta(q_1, b, A) = (q_1, \lambda)$ Dépiler A tant qu'on lit 'b'

– $\delta(q_1, \lambda, Z_0) = (q_1, \lambda)$ le mot est reconnu donc on vide la pile

2.4 Déterminisme

Le déterminisme avec un automate à pile est le même principe qu'avec un automate fini.

Un automate à pile est déterministe si

$$\left.\begin{array}{l}\forall q \in Q \\ \forall Z \in \Gamma \\ \forall x \in X\end{array}\right) \quad \begin{array}{c} Card(\delta(q, x, Z)) \leq 1 \ et \ card(\delta(q, \lambda, Z)) = 0 \\ Ou \ Card(\delta(q, \lambda, Z)) \leq \ 1 \ et \ card(\delta(q, x, Z)) = 0 \end{array}$$

R Il existe des langages qu'on ne peut reconnaître qu'avec un automate à pile non déterministe.

2.5 Automate à pile déduit d'une grammaire

– Soit L un langage hors contexte.
– Soit G tel que $L(G) = L$
– $G =< N, X, P, S >$
– $P = \{B \to \alpha, B \in N\alpha \in (NUX)^*$

Il existe un automate à pile A telque $L(A) = L(G) = L$ tel que
– La reocnnaissance se fait par pile vide $(F = \varnothing)$
– Il y a un seul était $Q = \{q\}; q_0 = q\}$
– Non déterministe
Il simule toutes les tentatives de la grammaire pour engendrer un mot de $L(G) = L(A) = L$.

$$\begin{array}{rcl} A & = & < Q, X, F, \Gamma, q_0, Z_0, \delta > \\ Q & = & \{q\} \ ; \ q_0 = q \ ; \ Z_0 = S \\ F & = & \varnothing \ ; \ \Gamma = NUX \end{array}$$

$$\delta \ tq \begin{cases} \delta(q, \lambda,) = \{(q, \alpha) \forall B \to \alpha P\} \\ \delta(q, x, x) = (q, \lambda) \forall x \in X \end{cases}$$

Chapitre 3

Machine de Turing

Langage régulier a^*b^*

Langage hors contexte – a^nb^n avec $n \leq 0$: Automate à pile déterministe
- $w_1.c.\widetilde{w_1}$ avec $n \leq 0$: Automate à pile déterministe
- $w_1.\widetilde{w_1}$ avec $n \leq 0$: Automate à pile non déterministe

Langage context sensitive $a^nb^nc^n$: Machine de Turing

La machine de Turing permet de conserver l'information plus longtemps qu'une pile, pour cela on utilise un ruban pour lire et conserver l'information écrite.

> R C'est le plus puissant des automates, bien que sont fonctionnement soit relativement simple.

3.1 Informellement

Alan Turing est un mathématicien qui invente en 1936 la machine de Turing dont le principe était simple : Posséder un ruban très grand, qu'il le considère d'une taille infini vers la droite [1], avec la possibilité de se déplacer vers la droite, vers la gauche d'écrire et de lire sur ce ruban.

> R Encore de nos jours, n'importe quel algorithme peut être résolu à l'aide d'une machine de Turing

3.2 Définition

Une machine de Turing est un 7-uplet : $< Q, X, F, \Gamma, B, q_0, \delta >$

Q Ensemble d'états ; $q_0 \in Q$ état initial

F Ensemble d'états accepteurs ; $F \subseteq Q$

X Alphabet(lecture)

Γ Alphabet de lecture et d'écriture

1. Celui-ci possède cependant une butée sur la gauche

B « Symbole blanc » que nous noterons #

δ Fonction de transition qui avec un état courant et le symbole lu donne le prochain état, le symbole écrit et le sens de déplacement $Q \times \delta \rightarrow Q \times \Gamma \times \{L, R\}$

3.3 Configuration

La configuration est un triplet (q, α_1, α_2).

q $q \in Q$, l'état courant.

α_1 Tout ce qui est avant la tête de lecture.$\alpha_1 \in \Gamma^*$

α_2 Tout ce qui est derrière la tête de lecture jusqu'au premier caractère blanc. $\alpha_2 \in \{\lambda\}$

<++>

3.4 Relation entre configuration

3.5 Langages acceptés ; langages décidés

3.6 Introduction à la calculabilité