Table des matières

Introduction

Ce mémoire entre le cadre de la théorie des langages et des automates et sont dédiés au développement de quelques produits des automates (produit en cascade, direct), ainsi que la décomposition en cascade d'automates. La décomposition d'un système complexe en éléments plus simples est l'objectif de toute science. Dans ce but nous allons étudier a travers ce travail la décomposition d'automates en composantes très simples, tels que les automates de permutations et les automates reset. Nous utilisons pour cette décomposition, le produit en cascade et le recouvrement, de la forme :

$$M \leq N_1\omega_1 N_2\omega_2 ...\omega_{n-1}N_n$$

tels que $N_1, N_2, ...N_n$ sont des automates simples, où ω_i est le produit en cascade.
La théorie d'automates est née de la convergence de plusieurs courants scientifiques. Le premier est issu des tentatives de logiciens tels que Church, Gödel ou Turing pour formaliser la notion de calcul et de machine. Cet effort a occupé toute la première moitié du vingtième siècle et pourtant, les automates finis, qui constituent le modèle le plus simple de machine, ne seront définis formellement que bien après les machines de Turing. Les systèmes dynamiques discrets forment la seconde source d'inspiration. Bien que leur étude remonte aux travaux de Morse datant de la première moitié du vingtième siècle, leurs liens avec les automates finis font encore à ce jour l'objet de recherches très actives. Le troisième courant, proche cousin du précédent, est la théorie de l'information bâtie par Shannon en 1948. Les problèmes de codage, étudiés notamment par Schûtzenberger dés les années cinquante, ont en effet profondément influencé la théorie des automates. La

quatrième source provient de linguistes tels que Chomsky qui, en cherchant à formaliser les langues naturelles, ont introduit les concepts de mots, langages, grammaires, que nous utilisons aujourd'hui.

Dans ce mémoire, on s'intéresse aux problème du produit en cascade d' automates, et leurs recouvrement et la décomposition des automates finis en automates simples tels que : automates reset et automates de permutations.

Ce mémoire se subdivise en trois chapitres.

Dans le premier chapitre, on donne un aperçu général sur le monoïde libre et les langages, puis on exposera des résultats concernant les automates finis et langages reconnaissables, dans la fin de ce chapitre, on donnera la définition d'automate minimal qui jouit de propriétés fort intéressentes, et en suite on donne la définition de monoïde syntaxique.

Le deuxième chapitre est consacré à l'étude du produit en cascade et le produit directe des semiautomates et leurs recouvrement. On divise cette étude en deux phases : la première phase sera consacrée aux produits des semiautomates, et la deuxième phase au recouvrement des semiautomates, et dans la fin de ce chapitre, on définit le recouvrement par permutation et reset semiautomate.

Dans le troisième chapitre, on étudie la décomposition en cascade d'un automate. Pour pouvoir définir le concept de décomposition en cascade d'automates, on doit d'abord définir qu'est-ce qu'un homomorphisme d'automates. Enfin on donne le théorème Krohn et Rhodes qui établit que tout automate fini peut être décomposer en automates simples qui sont : automates reset et automates de permutations, en utilisant le produit en cascade.

Chapitre 1

Généralités sur les langages et les Automates finis

Dans ce chapitre on donne un aperçu général sur les langages et les automates finis. Dans la première partie, on définit quelques notions de base tels que : semigroupe, monoïde libre, mots et langages. La deuxième partie est consacrée à l'étude de quelques propriétés des langages reconnaissables, ainsi que le calcul de l'automate minimal d'un langage donné.

1.1 Semigroupe, sous-semigroupe, monoïde

Définition 1.1.

Un *semigroupe* $(M, \cdot)$ est un ensemble M muni d'une opération binaire :

$$\cdot \ : M \times M \to M$$

Ce qui appelée le *produit* et notée $m \cdot n$ pour m et n éléments de M, et satisfait l'axiome d'associativité du produit :

$$\forall m_1, m_2, m_3 \in M, \quad (m_1.m_2) \cdot m_3 = m_1 \cdot (m_2 \cdot m_3).$$

Définition 1.2.

Soit $M^{'}$ une partie de M tel que :

$$\forall a, b \in M' : a \cdot b \in M'$$

donc M' est un sous-semigroupe de M.

Définition 1.3.

Un monoïde $(M, \cdot, 1_M)$ est un semigroupe avec un élément remarquable $1_M \in M$ qui a la propriété d'être un *élément neutre* pour le produit :

$$\forall m \in M, 1_M m = m = m 1_M.$$

1.2 Mots et langages

On considère maintenant un monoïde particulier $(\Sigma^*, \cdot)$, avec" $\cdot$ "est l'opération de concaténation :$(m \cdot n = mn)$. Il s'agit du *monoïde libre s*ur un ensemble Σ, c'est-à-dire le *monoïde engendré* par Σ, on le note Σ^*(i.e., les éléments de Σ^* ne sont pas liés entre eux par aucune relation). Lorsque Σ est un ensemble fini, on parle *d'alphabet,* et ses éléments sont appelés des *symboles* qu'on notera a,b,c etc...

Définition 1.4.

On définit *un mot* comme un élément du monoïde libre Σ^* dont le mot vide ε (mot de longueur nulle) est l'élément neutre de ce monoïde. Ou bien un mot est un séquence fini de symboles.

Remarques 1.5.

(1). La propriété fondamentale du monoïde libre est de pouvoir identifier les mots comme séquences de symboles, i.e., si on a $a_1.....a_n = b_1.....b_m$, alors $n = m$ et

$$\forall i \in \mathbb{N}, 0 < i \leq n, \;\; a_i = b_i. \;\; .$$

(2). Σ^*est dite la famille de tous les mots sur Σ.

(3). $\Sigma^+ = \Sigma^* - \{\varepsilon\}$.

Définition 1.6.

Un *langage* sur l'alphabet Σ est un ensemble de mots définis sur Σ, ou bien un langage est une partie de Σ^*.

Exemple 1.7.

(1). Le langage contenant uniquement le mot vide est $\{\varepsilon\}$,

(2). Le langage $L = \{xy \;/\; x \in \Sigma, y \in \Sigma\}$ est le langage des mots de deux lettres sur Σ.

(3). Considérons l'alphabet $\Sigma = \{a, b, c\}$. L'ensemble $\{a, aa, bbc, ccca, ababab\}$ est un langage fini.

1.3 Automates finis et langages reconnaissables

Définition 1.8.

Un *automate fini* M est défini par un quintuple $M = (Q, \Sigma, \delta, I, F)$, avec Q un ensemble fini d'états, Σ un alphabet fini, une fonction de transitions $\delta \subset (Q \times \Sigma \times Q)$ définie par $\delta : Q \times \Sigma \to Q$, un ensemble d'états *initiaux* $I \subset Q$ et un ensemble d'états acceptants $F \subset Q$.

Une *transition* de l'automate est un élément $(q, a, q') \in \delta$, et on dit que a est *l'étiquette* de la transition.

Définition 1.9.

Un *chemin* dans l'automate M est une séquence finie de transitions consécutives $(q_1, a_1, q_2)(q_2, a_2, q_3).....(q_n, a_n, q_{n+1})$, avec n la longueur du chemin. On appelle *étiquette* de ce chemin le mot $w = a_1....a_n$.

Définition 1.10.

Un mot $w = a_1.....a_n$ est dit *reconnu* par l'automate M, s'il existe un chemin étiqueté par w avec $q_1 \in I$ et $q_{n+1} \in F$. L'automate reconnaît ε s'il existe un état initial qui est aussi un état acceptant.

Le langage reconnu par l'automate M est l'ensemble des mots reconnus par M noté $L(M)$.

On dit que un langage L est régulier si ses mots sont *reconnus* par un automate fini (ensemble regulier).

Définition 1.11.

L'automate M est dit *complet* si :

$$\forall q \in Q, \forall a \in \Sigma, \exists\ p \in Q \text{ tel que : } \delta\,(q, a) = p.$$

Définition 1.12.

L'automate est dit *déterministe* lorsque l'ensemble des états initiaux est un singleton, $|I| = 1$ et $\forall q \in Q, \forall a \in \Sigma$, $|\{q' \ / \ \delta(q, a, q')\}| \leq 1$. Lorsque l'automate M est déterministe, on parle de DFA (deterministic finite automaton), sinon dans le cas le plus général, on dit que : c'est un NDFA (non-deterministic finite automaton). Les automates finis ont un graphe de transition étiquetées par des symboles, cependant on peut assez directement étendre les définitions pour avoir des automates étiquetés par des mots $(Q, \Sigma^*, \delta, I, F)$.

Définition 1.13.

On appelle langage *reconnaissable* sur Σ, un langage L reconnu par un automate fini M ($\exists\ M,\ L = L(M)$). La classe des langages reconnaissables sur un alphabet Σ sera notée $\mathrm{Rec}(\Sigma)$.

Remarque 1.14.

Les automates sont définis de manière formelle, avec des constructions ensemblistes, cependant on peut les représenter par leurs graphes pour aider l'intuition.

Exemple 1.15.

Soit l'automate défini par :

$(\{1,2,3\}, \{a,b,c\}, \{(1,a,2),(2,b,3),(3,c,2)\}, \{1\}, \{3\})$ sera représenté de la manière suivante telle que :

Les états sont représentés par des cercles. Une transition $(q,a,q') \in \delta$ est représentée par une flèche ayant pour origine q, pour destination q' et qui est annotée par l'étiquette a . Les états initiaux sont indiqués par flèche entrant, les états acceptants sont délimités par une des cercles en double -trait comme suite :

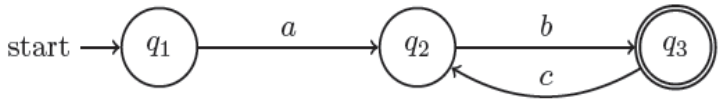

Figure 1.1. Un automate.

1.4 Automate minimal

Nous savons à présent qu'un langage est régulier si et seulement si il est accepté par un automate fini. cependant, plusieurs DFA peuvent accepter le même langage .

La question posée ici est de chercher parmi des automates équivalents, un automate qui serait, selon un sens encore à définir, *canonique.*

Il paraît naturel de vouloir minimiser le nombre d'états d'un DFA acceptant un langage régulier donné. En effet, lors de constructions comme le produit d'automates, il est préférable d'avoir peu d'états à traiter pour diminuer la taille de l'automate résultant. Nous allons montrer qu'à isomorphisme prés, il n'existe qu'un seul DFA acceptant un langage donné et possédant un nombre minimum d'états. Notons encore que la notion d'automate minimal peut être définie pour un langage quelconque.

1.4.1 Congruence syntaxique

Définition 1.16.

Soit $L \subseteq \Sigma^*$ un langage. Si w est un mot sur Σ, on dénote par $w^{-1} \cdot L$ l'ensemble des mots, qui est concaténés avec w, appartiennent à L i.e.,

$$w^{-1} \cdot L = \{u \in \Sigma^* / wu \in L\}.$$

On définit une relation sur Σ^*, notée $\sim_L$, de la manière suivante :

$$\forall x, y \in \Sigma^* x \sim_L y \Leftrightarrow x^{-1} \cdot L = y^{-1} \cdot L.$$

Proposition 1.17. [Rig09]

Soit $L \subseteq \Sigma^*$, et soit $\sim_L$ la relation d'équivalence définie ci-dessus. Il s'agit même d'une congruence à droite, i.e.,

$$\forall z \in \Sigma^*, \qquad x \sim_L y \Rightarrow xz \sim_L yz.$$

Remarque 1.18.

On parle souvent pour $\sim_L$ de la *congruence de Nerode.* On note $[w]_L$,
la classe d'équivalence du mot w par la relation $\sim_L$

$$[w]_L = \{u \in \Sigma^* / \ u \sim_L w\}.$$

Définition 1.19.

Soit $L \subseteq \Sigma^*$ un langage, on définit sur Σ^* la relation suivante :

pour $u, v \in \Sigma^*$:

$$u \equiv_L v \Leftrightarrow (\forall x, y \in \Sigma^* : xuy \in L \Leftrightarrow xvy \in L).$$

Il est facile de vérifier qu'il s'agit d'une relation d'équivalence sur Σ^*, et même d'une congruence (a droite et à gauche), i.e.,

$$\forall a \in \Sigma, u \equiv_L v \Rightarrow (ua \equiv_L va \text{ et } au \equiv_L av).$$

Remarque 1.20.

On parle souvent de la congruence syntaxique $\equiv_L$ et on dit que u, v sont syntaxiquement équivalents.

1.4.2 Automate minimal

Pour calculer l'automate minimal d'un langage donné L, on utilise la congruence de Nerode comme suit :

Définition 1.21.

On définit l'automate minimal $M_L = (Q_L, q_{0,L}, F_L, \Sigma, \delta_L)$, d'un langage $L \subseteq \Sigma^*$,
de la forme :

$$\begin{aligned}
&Q_L = \{w^{-1} \cdot L \ / \ w \in \Sigma\}, \\
&q_{0,L} = \varepsilon^{-1} \cdot L, \\
&F_L = \{w^{-1} \cdot L \ / \ w \in L\} = \{q \in Q_L \ / \ \varepsilon \in q\}, \\
&\delta_L(q, a) = a^{-1} \cdot q, \ \forall q \in Q_L, \ a \in \Sigma.
\end{aligned}$$

Lemme 1.22. [Rig09]

Soient $L \subseteq \Sigma^*$ un langage, et u, v deux mots sur Σ, on a :

$$(uv)^{-1} \cdot L = v^{-1} \cdot (u^{-1} \cdot L). \quad \blacksquare$$

Grâce au lemme précédent, la fonction de transition de l'automate
s'étend à $Q_L \times \Sigma^*$ par :

$$\delta_L(q, w) = w^{-1} \cdot q, \, \forall q \in Q_L, \, w \in \Sigma^*.$$

Remarque 1.23.

En vue de la définition de $\sim_L$, il est clair que l'ensemble des états de M :$\{w^{-1} \cdot L / w \in L\}$,
est en bijection avec l'ensemble quotient $\Sigma^* / \sim_L = \{[w]_L / w \in \Sigma^*\}$,
en effet, à chaque classe d'équivalence $[w]_L$ pour $\sim_L$ correspond un état $w^{-1} \cdot L$
de l'automate minimal M_L.

Exemple 1.24.

Soit le langage L définie par :

$$L = \{w \in \{a, b\}^* / \, |w|_a \equiv 0 (\mathrm{mod}\, 3)\} \text{ , Alors :}$$
$$abbaba \sim_L aaa \; car : abbaba^{-1} \cdot L = aaa^{-1} \cdot L = L$$
$$b \nsim_L ab \;\; car \; pour \; u = aa, bu \notin L, abu \in L$$

Il est facile de voir que pour L forme sur $\{a, b\}$ contenant un nombre de a multiple de 3. La congruence de Nerode possède trois classes : $[\varepsilon]_L$, $[a]_L$, $[aa]_L$. Dit autrement l'automate minimal A_L a trois états : $\varepsilon^{-1} \cdot L$, $a^{-1} \cdot L$, $(aa)^{-1} \cdot L$.
Pour définir la fonction de transition on a :

$\delta_L(\varepsilon^{-1} \cdot L, a) = a^{-1} \cdot (\varepsilon^{-1} \cdot L) = a^{-1} \cdot L$

$\delta_L(\varepsilon^{-1} \cdot L, b) = b^{-1} \cdot (\varepsilon^{-1} \cdot L) = b^{-1} \cdot L = \varepsilon^{-1} \cdot L \; car \; \varepsilon \sim_L b$

$\delta_L(a^{-1} \cdot L, a) = a^{-1} \cdot (a^{-1} \cdot L) = (aa)^{-1} \cdot L = L$

$\delta_L(a^{-1} \cdot L, b) = b^{-1} \cdot (a^{-1} \cdot L) = (ab)^{-1} \cdot L = a^{-1} \cdot L \quad car \;\; a \sim_L ab$

$\delta_L((aa)^{-1} \cdot L, a) = a^{-1} \cdot ((aa)^{-1} \cdot L) = (aaa)^{-1} \cdot L = \varepsilon^{-1} \cdot L \quad car \;\; \varepsilon \sim_L aaa$

$\delta_L((aa)^{-1} \cdot L, b) = b^{-1} \cdot ((aa)^{-1} \cdot L) = (aab)^{-1} \cdot L = (aa)^{-1} \cdot L \quad car \;\; a \sim_L aab$

Si on note $1, 2, 3$, les 3 langages $\varepsilon^{-1} \cdot L$, $a^{-1} \cdot L$, $(aa)^{-1} \cdot L$. On obtient l'automate minimal suivant :

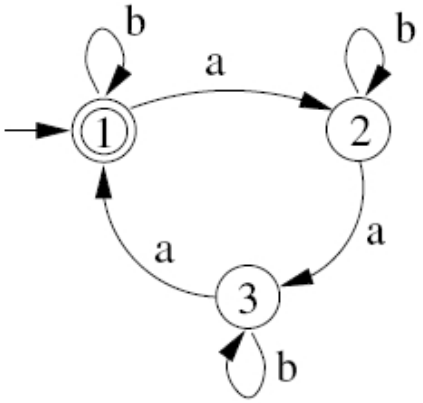

Figure.1.2. Un automate minimal.

1.4.3 Monoïde syntaxique

Définition 1.25.

Soit l'opération $\circ$ définie par :

$$\circ : \Sigma^*/\equiv_L \times \Sigma^*/\equiv_L \to \Sigma^*/\equiv_L : ([x], [y]) \to [x] \circ [y]$$

Définie par :

$$[x] \circ [y] = [z] \;\; si \;\; [x] \cdot [y] \subseteq [z] \quad ou \quad [x] \circ [y] = [xy]$$

où $\cdot$ représente l'opération de concaténation du mots.

Proposition 1.26.[Rig09]

Si on muni l'ensemble quotient $\Sigma^* / \equiv_L$ *par l'opération* $\circ$, *alors cette ensemble possède une structure de monoïde, tel que*

(1) le neutre $[\varepsilon]$:

$$\forall x \in \Sigma^* : [x] \circ [\varepsilon] = [x] \;\; .$$

(2) L'associativité :

$$\forall x, y, z \in \Sigma^* : ([x] \circ [y]) \circ [z] = [x] \circ ([y] \circ [z]) \;\; .$$

Définition 1.27.

Le monoïde $(\Sigma^* / \equiv_L, \circ)$ est le monoïde syntaxique de L.

Exemple 1.28.

Soit L : le langage formé des mots conprenant un nombre pair de a et un nombre pair de b, On obtient la table du monoïde syntaxique de L :

	ε	a	b	ab
ε	ε	a	b	ab
a	a	ε	ab	b
b	b	ab	ε	a
ab	ab	b	a	ε

Figure 1.3. Le monoide syntaxique de L.

Chapitre 2

Produit direct et produit en cascade des semiautomates et leurs recouvrement

Ce chapitre introduit le produit direct et le produit en cascade des semiautomates et leurs recouvrement. Dans la première partie on va étudier les produits des semiautomates, et la deuxième partie est consacrée à l'étude du recouvrement des semiautomates, en donnant quelques exemples, enfin on définit le recouvrement par permutation et reset semiautomates.

2.1 Produit direct et Produit en cascade des semiautomates

Définition 2.1.

Un semiautomate est le triple $M = (Q, \Sigma, F)$ tels que Q et Σ qui sont des ensembles finis et F est la fonction suivante : $F : Q \times \Sigma \rightarrow Q$.

Définition 2.2.

Soit $M = (Q, \Sigma, F)$ un semiautomate, considérons l'ensemble Σ^+ de tous les mots de longueur ≥ 1 de l'alphabet Σ. On définit une relation $\sim$ sur Σ^+ par :

$$\alpha \sim \beta \Leftrightarrow F_\alpha = F_\beta \quad \text{pour } \alpha, \beta \in \Sigma^+$$

tel que si $\alpha, \beta \in \Sigma$:

$$F_\alpha : Q \to Q \ \text{ défini par } qF_\alpha = F(q, \alpha) \ \ \forall q \in Q$$
$$F_\beta : Q \to Q \ \text{ défini par } qF_\beta = F(q, \beta) \ \ \forall q \in Q$$

cette relation est une relation d'équivalence, et on a $(\Sigma^+, \cdot)$ est un semigroupe avec l'opération de concaténation de mots "$\cdot$" de plus $\sim$ est une congruence sur Σ^+ c'est-a-dire si $\alpha, \beta, \gamma \in \Sigma^+$ et $\alpha \sim \beta$ alors $F_\alpha = F_\beta$ et

$$\forall q \in Q, qF_{\gamma\alpha} = qF_\gamma F_\alpha = (qF_\gamma)F_\alpha = (qF_\gamma)F_\beta = qF_{\gamma\beta}$$

et donc

$$F_{\gamma\alpha} = F_{\gamma\beta} \Leftrightarrow \gamma\alpha \sim \gamma\beta \text{ etc...}$$

On construit maintenant le semigroup quotient $(\Sigma^+ \setminus \sim, \cdot)$ ce semigroupe est dit le semigroupe de semiautomate M et notée $S(M)$, et les éléments de $S(M)$ sont les classes d'equivalences.

2.1.1 Produit direct

Définition 2.3.

Soient $M = (Q, \Sigma, F)$ et $M' = \left(Q', \Sigma', F'\right)$ deux semiautomates alors leurs **produit direct** est l'automate :

$$M \wedge M' = \left(Q \times Q', \Sigma, F \wedge F'\right) \text{ dans le cas } \Sigma = \Sigma'$$

Définie par :

$$\left(F \wedge F^{'}\right)\left(\left(q, q^{'}\right), \sigma\right) = \left(F\left(q, \sigma\right), F^{'}\left(q^{'}, \sigma\right)\right)$$

pour $\sigma \in \Sigma, \left(q, q^{'}\right) \in Q \times Q$

Exemple 2.4.

Soit M un semiautomate défini par :

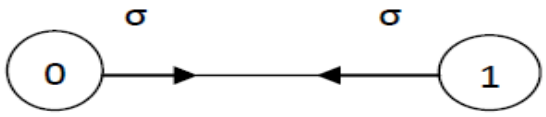

Figure 2.1. L'automate M.

Et soit $M^{'}$ un autre semiautomate défini par :

Figure 2.2. L'automate M'.

Le produit direct de M et M' est l'automate $M \wedge M^{'}$:

$$M \wedge M^{'} = (Q \times Q', \Sigma, F \Lambda F')$$

tel que

$Q \times Q' = \{(0,0), (0,1), (1,0), (1,1)\}$,

$\Sigma = \{\sigma\}$,

$F\Lambda F'\left((q,q'),\sigma\right)=\left(F\left(q,\sigma\right),F'\left(q',\sigma\right)\right),\ \sigma\in\Sigma,\ (q,q')\in Q\times Q',$

pour $(q,q')=(0,0):F\Lambda F'\left((0,0),\sigma\right)=\left(F\left(0,\sigma\right),F'\left(0,\sigma\right)\right)=(1,1),$

pour $(q,q')=(0,1):F\Lambda F'\left((0,1),\sigma\right)=\left(F\left(0,\sigma\right),F'\left(1,\sigma\right)\right)=(1,1),$

et de la même façon pour $(q,q')\in\{(1,0),(1,1)\}$

L'automate $M\wedge M^{'}$ est représenté de la forme :

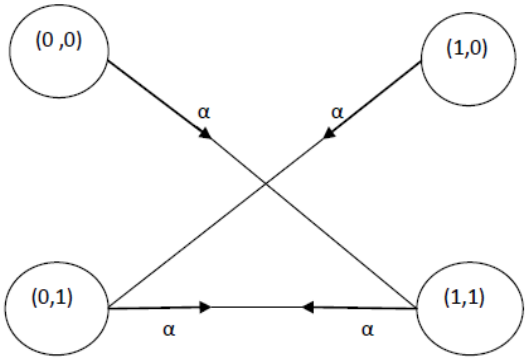

Figure 2.3. Le produit direct de M et M'

Définition 2.5.

Soient $M=(Q,\Sigma,F)$ et $M^{'}=\left(Q^{'},\Sigma^{'},F^{'}\right)$ deux semiautomates on définit leurs **produit direct général** dans le cas $\Sigma\neq\Sigma^{'}$ par : $M\times M^{'}=\left(Q\times Q^{'},\Sigma\times\Sigma^{'},F\times F^{'}\right)$ tels que,

$$\left(F\times F^{'}\right)\left(\left(q,q^{'}\right),\left(\sigma,\sigma^{'}\right)\right)=\left(F\left(q,\sigma\right),F^{'}\left(q^{'},\sigma^{'}\right)\right)$$

et $\sigma\in\Sigma,\sigma^{'}\in\Sigma^{'},\left(q,q^{'}\right)\in Q\times Q^{'}.$

Exemple 2.6.

Soit M un semiautomate défini par :

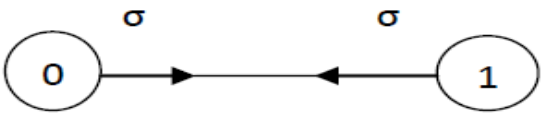

Figure 2.4. L'automate M.

Et soit M' un autre semiautomate défini par :

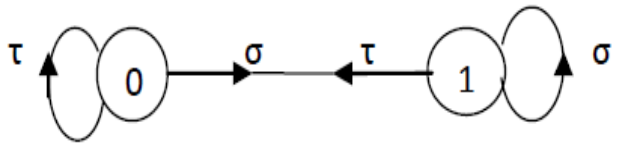

Figure 2.5. L'automate M.

Le produit direct général de M et M' est l'automate $M \times M^{'}$ défini par :

$$M \times M^{'} = \left(Q \times Q^{'}, \Sigma \times \Sigma^{'}, F \times F^{'}\right)$$

tels que

$Q \times Q' = \{(0,0), (0,1), (1,0), (1,1)\}$,

$\Sigma \times \Sigma' = \{(\sigma,\sigma), (\sigma,r)\}$,

$F \times F'((q,q'),(\sigma,\sigma')) = (F(q,\sigma), F'(q',\sigma'))$, $(\sigma,\sigma') \in \Sigma$, $(q,q') \in Q \times Q'$,

pour $(q,q') = (0,0) : F \times F'((0,0),(\sigma,\sigma)) = (F(0,\sigma), F'(0,\sigma)) = (1,1)$,

$F \times F'((0,0),(\sigma,r)) = (F(0,\sigma), F'(0,r)) = (1,0)$,

et de la même façon pour $(q,q') \in \{(0,1),(1,0),(1,1)\}$

L'automate $M \times M^{'}$ est représenté de la forme :

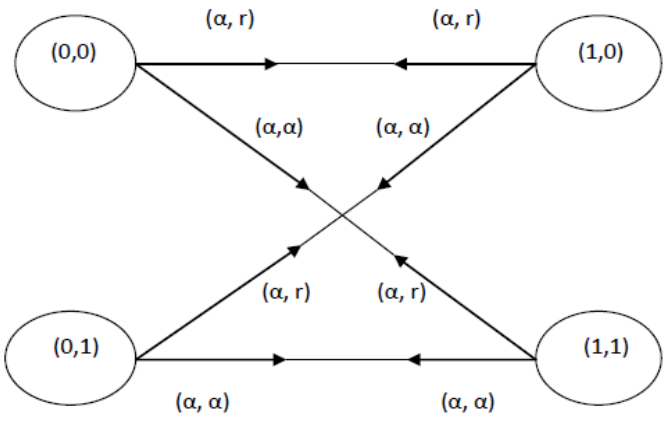

Figure 2.6. Le produit direct général de M et M'

2.1.2 Produit en cascade

Définition 2.7.

Soient $M = (Q, \Sigma, F)$ et $M' = \left(Q', \Sigma', F'\right)$ deux semiautomates on va définir leurs **produit en cascade** par : $M\omega M' = \left(Q \times Q', \Sigma', F^{\omega}\right)$ avec $w : Q' \times \Sigma' \to \Sigma$

tel que

$$F^{\omega}\left(\left(q, q'\right), \sigma'\right) = \left(F\left(q, w\left(q', \sigma'\right)\right), F'\left(q', \sigma'\right)\right)$$

pour $\sigma' \in \Sigma'$, $\left(q, q'\right) \in Q \times Q'$.

Exemple 2.8.

Soit M un semiautomate défini par :

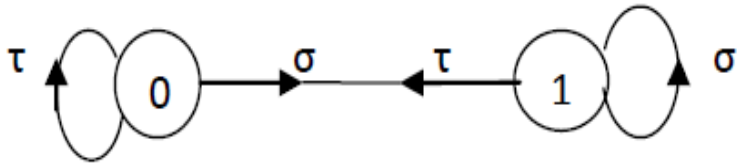

Figure 2.7. L'automate M.

Et Soit M' un semiautomate défini par :

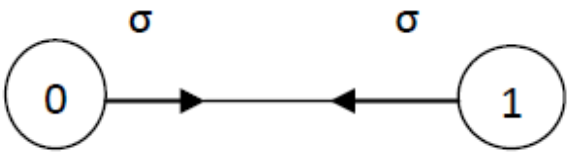

Figure 2.8. L'automate M'.

Le produit en cascade de M et M' est l'automate $M\omega M^{'}$ défini par :

$$M\omega M^{'} = \left(Q \times Q^{'}, \Sigma^{'}, F^{\omega}\right)$$

tels que

$Q \times Q' = \{(0,0), (0,1), (1,0), (1,1)\}$,

$\Sigma' = \{\sigma\}$, $\Sigma = \{\sigma, r\}$, $Q' = \{0,1\}$

$w : Q' \times \Sigma' \to Q$, défini par $w(0,\sigma) = \sigma, w(1,\sigma) = r$,

$F^{\omega}\left(\left(q, q^{'}\right), \sigma^{'}\right) = \left(F\left(q, w\left(q^{'}, \sigma^{'}\right)\right), F'\left(q^{'}, \sigma^{'}\right)\right)$

pour $(q, q') = (0,0) : F^{\omega}((0,0), \sigma) = \left(F(0, w(0,\sigma)), F'(0,\sigma)\right)$

$= (F(0,\sigma), F'(0,\sigma)) = (1,1).$

et de la même façon pour $(q, q') \in \{(0,1), (1,0), (1,1)\}$

L'automate $M\omega M^{'}$ est représenté de la forme :

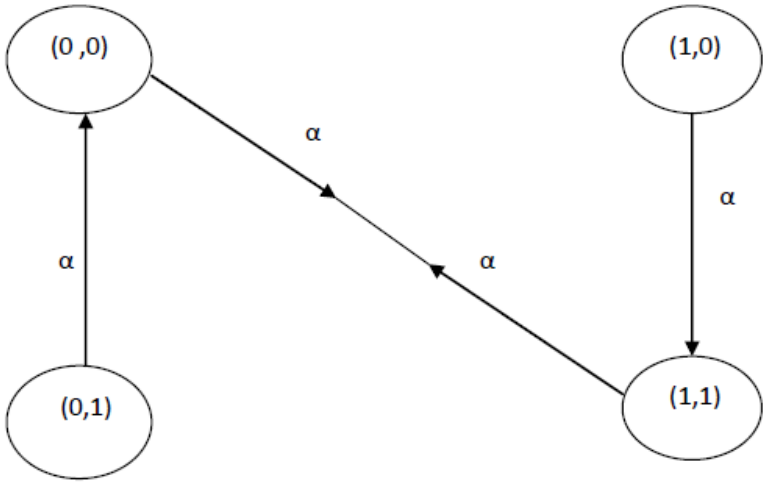

Figure 1.9. Le produit en cascade de M et M'

Remarque 2.9.

Ce type de produit peut être généralisé, donc on peut définir le produit en cascade de n automates. Plus formellement, on a la définition suivante du produit en cascades, tirée de l'article de Maler et Pnueli [MP94]. Ces auteurs ont réécrit plusieurs définitions et théorèmes concernant les semi-groupes en terme d'automates. Le traitement étant intéressant, nous l'emploierons dans ce qui suit.

Définition 2.10.

Soit $\mathcal{B}_i = (Q_1 \times ... \times Q_{i-1} \times \Sigma, Q_i, F_i)$ une famille d'automates ayant comme alphabets d'entrée les produits $Q_1 \times ... \times Q_{i-1} \times \Sigma$ et ayant F_i comme fonctions de transition. Le produit en cascade

$$C = (\Sigma, Q, F') = \mathcal{B}_1 \omega \mathcal{B}_2 ... \omega \mathcal{B}_k$$

est un automate tel que :

(1) $Q = Q_1 \times Q_2 \times ... Q_k$.

(2) La fonction de transition globale δ', est évaluée coordonnée par coordonnée par

$F'(q_1...q_k, \sigma) = (F_1(q_1, \sigma), F_2(q_2, (q_1, \sigma)), ..., F_k(q_k, (q_1...q_{k-1}, \sigma)))$

2.2 Recouvrement

2.2.1 Recouvrement d'un semiautomate

Définition 2.11.

Soit $M = (Q, \Sigma, F)$ un semiautomate complet et $\forall q \in Q, \forall \alpha \in \Sigma^*$ on définit, la fonction $F_\alpha : Q \to Q$ par :

$qF_\alpha = F(q, \alpha) \quad \forall\, q \in Q$

Soient $M = (Q, \Sigma, F), M' = \left(Q', \Sigma', F'\right)$ deux semiautomates, si $\zeta : \Sigma \to \Sigma'$ est une fonction et $\eta : Q' \to Q$ est une fonction tel que :

$$\eta\left(q'\right) F_\alpha \subseteq \eta\left(q' F'_{\xi(\alpha)}\right) \quad \forall q \in Q \text{ et } \alpha \in \Sigma^*$$

On dit que (η, ζ) est un **recouvrement** de M par M' est noté $M \leq M'$.

Exemple 2.12.

Soit $M = (Q, \Sigma, F)$ un semiautomate, on définit la relation $\sim$ sur Σ par :

$$\sigma \sim \sigma_1 \Leftrightarrow F_\sigma = F_{\sigma_1} \quad \forall \sigma, \sigma_1 \in \Sigma$$

Considérons le semiautomate $M' = \left(Q, \Sigma / \sim, \overline{F}\right)$ défini par : $\overline{F}\,(q, [\sigma]) = F\,(q, \sigma)$ pour $q \in Q$ et $[\sigma] \in \Sigma \backslash \sim$.

Maintenant on dit que ;

$\zeta : \Sigma \to \Sigma \backslash \sim$ défini par : $\zeta\,(\sigma) = [\sigma]$ pour $\sigma \in \Sigma$

$\eta : Q \to Q$ défini par $\eta\,(q) = q$ pour $q \in Q$

et on obtient le recouvrement de M par M'.

Exemple 2.13.

Soit M un semiautomate défini par le diagramme :

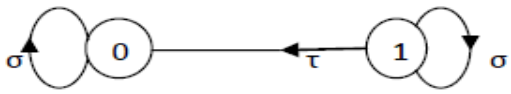

Figure.2.10. L'automate M

Et M'défini par :

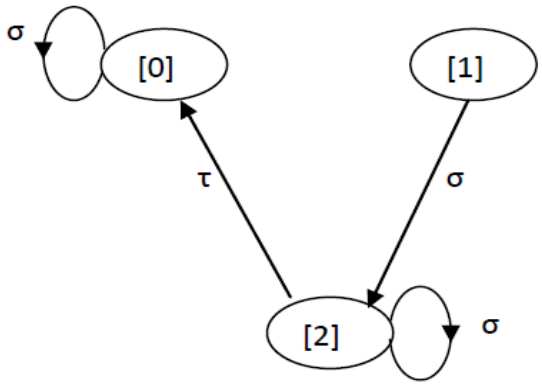

Figure.2.11. L'automate M'.

On définit :$\eta : [0] \to 0$, $[1] \to 1$, $(\eta, 1_\Sigma)$ ne donne pas un recouvrement, car :$\eta\,([1])\,F_r = 0 \subsetneq \eta\,([1]\,F'_r) = \varnothing$ mais $\eta : [0] \to 0, [2] \to 1$, $(\eta, 1_\Sigma)$ donné un recouvrement $M \leq M'$.

2.2.2 Recouvrement par permutation et reset semiautomates

Une classe importante de semiautomates sont les permutation-reset semiautomates qui sont définis comme suit :

Définition 2.16.

Soit M un automate, on définit trois types particuliers de transitions dans M :

(1). Un événement a de l'alphabet d'entrée Σ de M est appelé un reset si la fonction induite par a sur les états de M une constante :

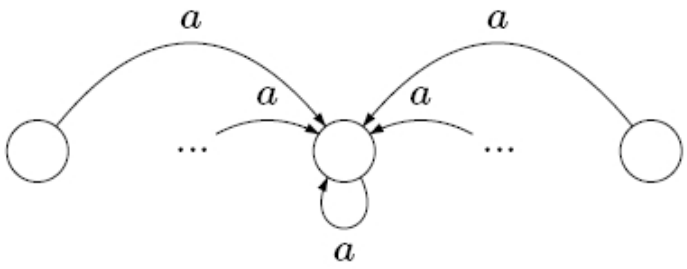

Figure.2.12. Un reset.

(**2).** Un événement a de l'alphabet d'entrée Σ de M est appelé une identité si la fonction induite par a sur les états de M est une fonction identité :

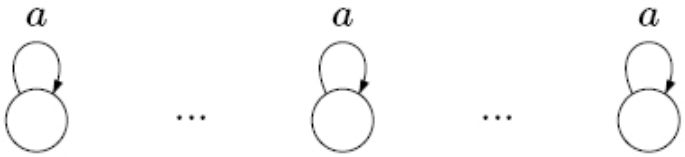

Figure.2.13. Une identité.

(**3).** Un événement a de l'alphabet d'entrée Σ de M est appelé une permutation si la fonction induite par a sur les états de M permute l'ensemble des états sur lesquels a est définie :

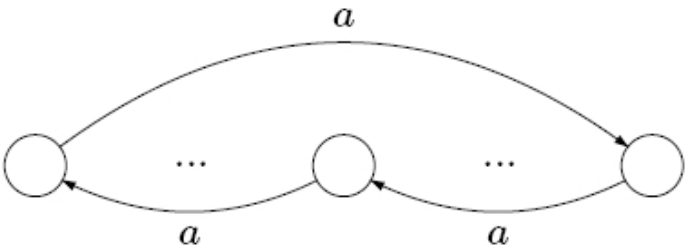

Figure.2.14. Une permutation.

Définition 2.17.

Un automate reset est un automate dans lequel toutes les lettres $\sigma \in \Sigma$ induisent des resets ou des identités. Un automate permutation-reset est un automate dans lequel chaque lettre σ de Σ induit soit une permutation, soit un reset sur les états sur lesquels σ est définie.

Théorème 2.18. [Hol82]

Soit M une permutation-reset machine, alors $M \leq N^{\cdot}\omega P$.

Tel que $N^{\cdot}$ est un semiautomate reset et P est un semiautomate de permutation.

Preuve.

Soit $M = (Q, \Sigma, F)$ un semiautomate, supposons $\Theta = \{\sigma \in \Sigma \setminus \ (Q)F_\sigma = Q\}$

et $\Xi = \{\sigma \in \Sigma \setminus \ \mid (Q)F_\sigma| = 1\}$.

On définit G le sous-semigroupe de $S(M)$ engendré par Θ et supposons que

$p = (G, \Sigma, \overline{F}\ )$ tel que,

$$[\alpha]\,\overline{F}_\alpha = [\alpha\theta] \quad \text{pour } \theta \in \Theta, \alpha \in \Theta^*$$
$$[\alpha]\,\overline{F}_\xi = [\alpha] \quad \text{pour } \xi \in \Xi, \alpha \in \Theta^*$$

Soit $N = (Q, G \times \Sigma, F^*)$ défini par

$$qF^*_{(g,\xi)} = qF_\alpha F_\xi \left(F_\alpha\right)^{-1}$$

Telle que $g = [\alpha] \in G, \alpha \in \Theta^*, \xi \in \Xi$ et $q \in Q$.

Maintenant F_α est une permutation de Q , donc $(F_\alpha)^{-1}$ est défini , en autre

$$\left|(Q)\,F^*_{(g,\xi)}\right| = \left|(Q)\,F_\alpha F_\xi\,(F_\alpha)^{-1}\right| = 1 \text{ pour } (Q)\,F = Q \text{ et } |(Q)\,F_\xi| = 1.$$

et N est un reset semiautomate, le semiautomate $N^{\cdot}$défini par : $N^{\cdot} = (Q, (G \times \Sigma) \cup \{\Lambda\}, F^{**})$ tel que,

$$qF^{**}_{(g,\xi)} = qF^*_{(g,\xi)} \text{ pour } q \in Q, g \in G, \xi \in \Xi \text{ et } qF^{**}_\Lambda = q \text{ pour } q \in Q.$$

Maintenant, on définit $w : G \times \Sigma \to G \times \Sigma \cup \{\Lambda\}$ par :

$$w(g,\sigma) = \begin{cases} \Lambda \text{ si } \sigma \in \Theta \\ (g,\sigma) \text{ si } \sigma \in \Xi \end{cases}$$

Nous formons le produit $N \cdot \omega P$ tel que l'application du produit en cascade $N \cdot \omega P$ et notée par F^{ω}.

L'application de recouvrement $\phi : Q \times G \to Q$ défini par :

$$\phi(q,g) = qF_{\alpha} \text{ tel que } g = [\alpha] \in G, q \in Q.$$

Il faut Maintenant déterminer les proprietés de recouvrement pour ϕ . ϕ est surjective car $\phi \neq \varnothing$ et F_{α} est une permutation de Q, donc soit $\sigma \in \Theta$ et $(q,g) \in Q \times G$,si

$$g = [\alpha], \alpha \in \Theta^{*} \text{ alors } (\phi(q,[\alpha]))\, F_{\sigma} = (qF_{\alpha})\, F_{\sigma} = qF_{\alpha\sigma} = \phi(q,[\alpha\sigma]) \text{ pour } \alpha\sigma \in \Theta^{*}$$

donc

$$\begin{aligned} \phi((q,[\alpha])\, F_{\alpha}^{\omega}) &= \phi\left(qF^{*}_{([\alpha],\sigma)}, [\alpha]\, \overline{F}_{\sigma}\right) \\ &= \phi(q,[\alpha\sigma]) \end{aligned}$$

si $\sigma \in \Xi$ et $(q,g) \in Q \times G$ avec $g = [\alpha]$ pour $\alpha \in \Theta^{*}$ alors

$$(\phi(q,[\alpha]))\, F_{\alpha} = (qF_{\alpha})\, F_{\sigma} = qF_{\sigma}$$

sinon

$$\begin{aligned} \phi((q,[\alpha])\, F_{\sigma}^{\omega} &= \phi\left(qF^{**}_{([\alpha,\sigma])}, [\alpha]\, \overline{F_{\sigma}}\right) \\ &= \phi\left(qF^{*}_{([\alpha],\sigma)}, [\alpha]\right) \\ &= \phi\left(qF_{\alpha}F_{\sigma}(F_{\alpha})^{-1}, [\alpha]\right) \\ &= qF_{\alpha}F_{\sigma}(F_{\alpha})^{-1}F_{\alpha} \\ &= qF_{\alpha}F_{\sigma} = qF_{\alpha} \end{aligned}$$

donc $(\phi(q,[\alpha])) \subseteq \phi((q,[\alpha])\, F_{\sigma}^{\omega})$.■

Chapitre 3

Décomposition d'automates

La décomposition est la simplification de tout système complexe, pour cela on étudie dans ce chapitre la décomposition d'automates dont les composantes sont très simples. Dans le chapitre précédent on a déterminé les automates et leurs produits et leurs recouvrement. Donc la décomposition d'automates est de la forme : $M \leq N_1\omega_1 N_2\omega_2 ...\omega_{n-1}N_n$ tels que $N_1, N_2, ...N_n$ sont des automates simples, où ω_i est le produit en cascade.

3.1 La décomposition en cascade d'automates

Dans cette section, nous présentons le concept de décomposition en cascade d'automates.

Pour pouvoir définir le concept de décomposition en cascade d'automates, il faut d'abord définir ce qui est un homomorphisme d'automates.

Définition 3.1.

Une fonction surjective $\Phi : Q \to Q'$ est un homomorphisme d'automates, de $M = (\Sigma, Q, F)$ sur $M' = (\Sigma, Q', F')$, si elle satisfait l'égalité suivante,

$$\forall q \in Q \text{ et } \forall \sigma \in \Sigma : \Phi(F(q, \sigma)) = F'(\Phi(q), \sigma).$$

On dit que l'homomorphisme d'automates est *partiel,* lorsqu'il n'est défini que sur un sous-ensemble de Q.

Exemple 3.2.

Soient M et M' les automates sur l'alphabet $\Sigma = \{a, b\}$:

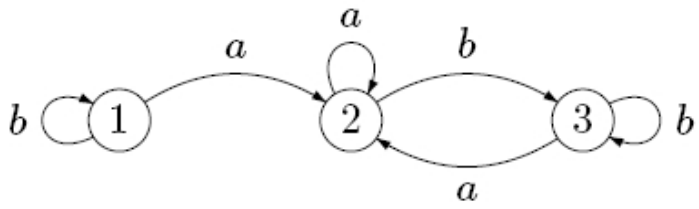

Figure 3.1. l'automate M

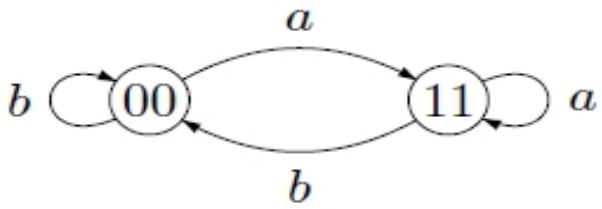

Figure 3.2. l'automate M'

L'homomorphisme $\Phi : Q \setminus \{1\} \rightarrow Q'$ défini par : $\Phi(2) = 11$ et $\Phi(3) = 00$, est un homomorphisme partiel d'automates car :

$$\Phi(F(2,a)) = \Phi(2) = 11 = F'(11,a) = F'(\Phi(2),a),$$
$$\Phi(F(2,b)) = \Phi(3) = 00 = F'(11,b) = F'(\Phi(2),b),$$
$$\Phi(F(3,a)) = \Phi(2) = 11 = F'(00,a) = F'(\Phi(3),a),$$
$$\Phi(F(3,b)) = \Phi(3) = 00 = F'(00,b) = F'(\Phi(3),b).$$

et que lorsque $q = 1$, la fonction Φ n'est pas définie.

La décomposition en cascade d'un automate M peut alors être définie comme suit :

Définition 3.3.

Soit M un automate. Une décomposition en cascade de M est donnée par un produit en cascade $C = N_1 \omega N_2 ... \omega N_k$, $k > 1$, et un homomorphisme (possiblement partiel) Φ de C sur M.

Exemple 3.4.

Soit l'automate M suivant :

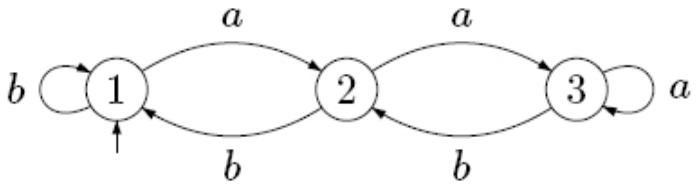

Figure 3.3. L'automate M

Cet automate admet la décomposition suivante avec l'homomorphisme

$$\Phi(0,0) = 1, \quad \Phi(0,1) = 2$$
$$\Phi(1,0) = 2, \quad \Phi(1,1) = 3.$$

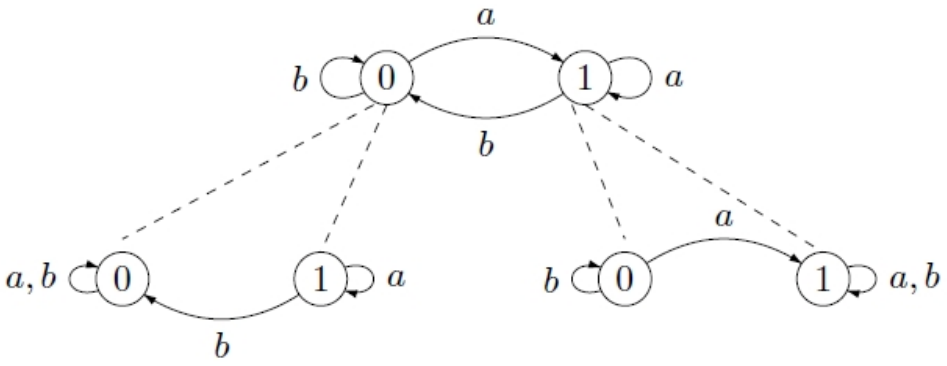

Figure.3.4. Une décomposition en cascade

Un théorème de Krohn et Rhodes [KR63],[KR82] dit que tout automate admet une décomposition dont les composantes sont très simples.

3.2 Théorème de K.B.Krohn et J.L.Rhodes

On peut couvrir tout semiautomate M par le produit directe et le produit en cascade de semiautomate en deux types

(1) grouplike simple semiautomate avec les groupes simples qui sont des images homomorphes du sous-groupe semiautomate du G_M de M.

(2) deux semiautomates de reset.

Théorème 3.5. [Gin68] *Le grouplike semiautomate employé pour couvrir M a été obtenu en trois étapes :*

(1) M a été couvert un produit en cascade de semiautomate de permutation-reset.

(2) Chaque semiautomate de permutation-reset a été couvert par le produit en cascade de semiautomate de permutation et semiautomate de reset.

(3) Le semiautomate de permutation obtenu a été couvert par le produit en cascade du semiautomat simple de grouplike.

Preuve : voire [Gin68].

Bibliographie

[Eil(67)] S.Eilenberg, *Automata, Languages and Machines, Vol A,* Academic Press ,INC (1976)

[Hol82] W.M.L.Holcombe, *Algebraic automata theory,* Cambridge University (1982).

[Gin68] A.Ginzburg, *Algebraic theory of automata,*Academic press , New york. London (1968).

[Pin06] Jean-Eric Pin, *Automates finis,* (2006).

[KR63] K.Krohn et J.Rhodes,*Algebraic Theory of Machines.Proc.Symp.Math. Theory of Automata, Brooklyn, N.Y.,*
(1962) pp. 341-384. Wiley,New York,(1963).

[KR82] K.Krohn et J.Rhodes, *Algebraic Theory of Machines. I. Prime Decomposition Theorem for Finite Semigroups and* , Trans. Am. Math. Soc., 116, pp.450-464 (1965).

[MP94] O.Maler et A.Pnueli, *On the Cascaded Decomposition of Automata, its Complexity and its Application to Logic* , document non publié, 48 pages (1994).

[Rig09] M.Rigo, *Théorie des automates et langages formels,*Université de liège (2009).

[Wol01] P.Wolper, *Introduction à la calculabilité,* Dunod ,Paris,(2001-2006).

[Zei64] H.P.Zeiger, *Loop-free* synthesis *of Finite-State Machines.Ph.D.* Thesis Elec. Eng. Department, M.I.T.,(1964)

[Zei67] H.P.Zeiger, *Cascade Synthesis of Finite-State Machines* , *Information and Control*, 10, pp. 419-433 (1967).