ELSEVIER

Theoretical Computer Science 242 (2000) 361–376

Theoretical Computer Science

www.elsevier.com/locate/tcs

Families of locally testable languages

Pascal Caron *

Laboratoire d'Informatique Fondamentale et Appliquée de Rouen, Université de Rouen, F-76821 Mont-Saint Aignan, Cédex, France

Received May 1997; revised November 1998
Communicated by M. Nivat

Abstract

Kim, McNaughton and McCloskey have produced a polynomial time algorithm in order to test if a deterministic automaton recognizes a locally testable language. We provide a characterization in terms of automata for the strictly locally testable languages and for the strongly locally testable languages, two subclasses of locally testable languages. These two characterizations lead us to polynomial time algorithms for testing these families of languages.

Keywords: Algorithms; Automata; Locally testable languages; Regular languages

1. Introduction

The local testability concept can be illustrated owing to a window and three sets. The window is slided on the input word without considering number nor order of factors appearing in it. This window is equipped with a set of prefixes, a set of suffixes, and a set of factors. Prefixes and suffixes are smaller than the size of the window. Factors have exactly the same size as the window. This mechanism allows us to define three families of languages. The first one, the family of strictly locally testable languages, is defined as follows: whether or not a word is in the language depends on the fact that its prefixes, suffixes and factors belong to the sets previously defined. The second one, the family of locally testable languages, is defined owing to a set of sliding windows. The last one, the family of strongly locally testable languages, is defined as the previous family without considering the set of prefixes and the set of suffixes. The locally testable languages have been discovered independently by Brzozowski and Simon [5] and McNaughton [12]. Kim et al. [9] have given a polynomial time algorithm to recognize locally testable automata. Beauquier and Pin [3] have provided an algebraic

* E-mail: caron@dir.univ-roven.fr.

0304-3975/00/$ - see front matter
PII: S0304-3975(98)00332-6

property of strongly locally testable languages. This paper studies a characterization of these languages in terms of automata. We also generalize the notion of local automaton which has been introduced by Berstel and Pin [4] and we show the equivalence between languages recognized by such an automaton and the strictly locally testable languages.

This paper has five sections including this one. The second section introduces notations and usual definitions on the various families of locally testable languages, on semigroups and on automata. The following two sections are devoted to the characterization in terms of automata of both strictly locally testable languages and strongly locally testable languages. The last section presents new algorithms deduced from these characterizations as well as their complexity. The paper closes with a discussion on a new problem.

2. Notation and terminology

In this section we recall some definitions and establish notation. For a finite set A, $|A|$ denotes the cardinality of A. Let Σ be a finite non-empty alphabet. Σ^* denotes the set of all words over Σ and Σ^+ the set of all non-empty words over Σ. Elements of Σ^+ will be denoted by $a_1 \dots a_n$, $a_i \in \Sigma$. For $w \in \Sigma^+$, $|w|$ will denote the length of w. An interior factor of a word w is a factor which is neither a prefix nor a suffix of w. A language is any subset of Σ^*. Throughout this paper, L will denote a non-empty regular language.

2.1. Automata

Notation used in this paper follows the book of Hopcroft and Ullman [8]. Our two characterizations are described on the state transition graph of a deterministic minimal automaton $\mathcal{M} = (\Sigma, Q, i, F, \delta)$. We shall use the term connected component (CC) to refer to any subgraph whose underlying undirected graph is connected. By SCC we mean a strongly connected component. A SCC C_1 is an ancestor (resp. descendant) of an SCC C_2 if there is a path from C_1 to C_2 (resp. from C_2 to C_1). The reaching component from C_1 to C_2 denoted by $C_{1,2}$ is the connected component including C_1 and C_2 as well as all the paths beginning in C_1 and terminating in C_2.

2.2. Semigroups

We introduce here some needful definitions for a good understanding of the paper. The reader can refer usefully to the books of Lallement [10] and Pin [17] for an overview of this domain.

Recall that a semigroup S is a set equipped with an associative multiplication. All semigroups considered in this paper are finite semigroups except for the free semigroup. An element e of a semigroup is an idempotent if $e^2 = e$. We denote by $E(S)$ the set of idempotents of the semigroup S. A monoid is a semigroup with an identity. Let S be a semigroup. We denote by S^1 the monoid equal to S if S has an identity and

$S \cup \{1\}$ otherwise. Let S and T be two semigroups. A semigroup morphism $\varphi : S \to T$ is a function from S into T such that, for every s, s' in S, $\varphi(s)\varphi(s') = \varphi(ss')$. Recall the definition of Green relations $\mathscr{R}$, $\mathscr{L}$ and $\mathscr{D}$:
$s\mathscr{R}t$ if and only if there exists $a, b \in S^1$ such that $sa = t$ and $tb = s$.
$s\mathscr{L}t$ if and only if there exists $a, b \in S^1$ such that $as = t$ and $bt = s$.
$s\mathscr{D}t$ if and only if there exists $a \in S^1$ such that $s\mathscr{R}a$ and $a\mathscr{L}t$.

We will use the following lemma in the characterization of strongly locally testable languages.

Lemma 2.1 (Beauquier and Pin [3]). *Let S and T be two finite semigroups. Let $\pi : S \to T$ be a surjective morphism. Let t and t' be two elements of T such that $t\mathscr{R}t'$ (respectively $t\mathscr{L}t'$, $t\mathscr{D}t'$). Then there exist s and s' such that $\pi(s) = t$, $\pi(s') = t'$ and $s\mathscr{R}s'$ (respectively $s\mathscr{L}s'$, $s\mathscr{D}s'$).*

2.3. Languages

2.3.1. Strictly locally testable languages

The concept of strict local testability can be seen as a computational mechanism on three sets of words. A window of size k scans the input word to verify whether its prefix is in the first set, its factors are in the second set and its suffix is in the third set. We recall here the definition of strictly k-testable languages. This definition is used by Pin in his Ph.D. thesis [16] and by De Luca and Restivo [11].

Definition 2.2. A language $L \subset \Sigma^+$ is strictly k-testable if there exist a positive integer k and four sets U, V, W, F with $F, U, V \subset \Sigma \cup \Sigma^2 \cup \Sigma^3 \cdots \cup \Sigma^{k-1}$ and $W \subset \Sigma^k$ such that

$$L = [(U\Sigma^* \cap \Sigma^* V) \backslash \Sigma^* W \Sigma^*] \cup F.$$

There exists another definition of strictly k-testable languages. This one is given by McNaughton and Papert [13].

Definition 2.3. Let k be a positive integer. For $w \in \Sigma^+$ of length $\geqslant k$, let $L_k(w)$, $R_k(w)$ and $I_k(w)$ be respectively the prefix of length k, the suffix of length k and the set of interior factors of length k of the word w. A language $L \subseteq \Sigma^*$ is strictly k-testable if and only if there exist three sets X, Y, Z of words on Σ such that for all $w \in \Sigma^+$, $|w| \geqslant k$, $w \in L$ if and only if $L_k(w) \in X$, $R_k(w) \in Y$ and $I_k(w) \subseteq Z$.

If $|w| = k$ then $L_k(w) = R_k(w) = w$. If $|w| = k$ or $|w| = k + 1$, then $I_k(w) = \emptyset$.

Notice that these two definitions are not equivalent, the language $ab\Sigma^*$ is strictly 2-testable according to Definition 2.3 and not according to Definition 2.2, but they lead to the same definition of strictly locally testable languages. A language is *strictly locally testable* if it is strictly k-testable for some $k > 0$. In order to give the syntaxic characterization of strictly locally testable languages, we need to introduce the notion of constant defined by Schützenberger [18]. Let S be the syntaxic semigroup of a language L. An

element c of S is a *constant* for L if for all $p,q,r,s \in S^1$, $pcq, rcs \in L \Rightarrow pcs \in L$. This notion of constant allows us to state the characterization theorem due to De Luca and Restivo [11].

Theorem 2.4. *A language L is strictly locally testable if and only if there exists a positive integer k such that every word of length $\geqslant k$ is a constant for L.*

2.3.2. Locally testable languages

A language L is locally testable if, taking two words having the same prefix, suffix and factors, both are in L or neither is in L. A more formal definition is proposed by Zalcstein [19]. A *k-testable language* is a boolean combination of strictly k-testable languages. A language is *locally testable* if it is k-testable for some $k > 0$.

The proposition we state here is taken as a definition in [13]. It is also used in [15]. Let L be a k-testable language. Let u and v be two words of Σ^* such that u and v have the same prefixes of length k, the same suffixes of length k and the same set of interior factors of length k. Then we have: $u \in L \Leftrightarrow v \in L$

The characterization given by Brzozowski, Simon [5] and McNaughton [12] for locally testable languages is formulated in terms of semigroups.

Theorem 2.5. *Let L be a recognizable language on Σ^+. The following conditions are equivalent:*

1. *L is locally testable.*
2. *The syntaxic semigroup S of L is locally idempotent and commutative, i.e. for all $e \in E(S)$, eSe is idempotent and commutative.*

This first characterisation leads to an exponential polynomial time algorithm. Kim et al. [9] describe a polynomial time algorithm allowing us to decide, having a deterministic minimal automaton, whether it recognizes a locally testable language.

Recall here the set of definitions and the theorem on which this algorithm is based. Let $\mathcal{M} = (\Sigma, Q, i, F, \delta)$ be an automaton. A strongly connected component (SCC) C_1 of the state transition graph of $\mathcal{M}$ is *s-local* if and only if there do not exist two distinct states p and q in C_1 and a word w in Σ^+ such that $\delta(p,w) = p$ and $\delta(q,w) = q$. Let C_1 and C_2 be two disjoint SCC; then C_1 and C_2 are *pairwise s-local* if and only if there do not exist two distinct states p and q, respectively, in C_1 and C_2 and a word $w \in \Sigma^+$ such that $\delta(p,w) = p$ and $\delta(q,w) = q$. Let C be a CC of the state transition graph of $\mathcal{M}$. The *transition span* in C of a state $p \in Q$ is the set $TS(C,p) = \{x, x \in \Sigma^* \mid$ for every prefix w of x, $\delta(p,w)$ is in $C\}$. The states p and q are *TS-equivalent* in C if and only if $TS(C,p) = TS(C,q)$. Let C_j be an SCC of the state transition graph of a deterministic automaton. The graph is *TS-local w.r.t.* C_j if and only if it has the following property, for every SCC C_i which is an ancestor of C_j, either: (1) C_i and C_j are pairwise s-local, or otherwise (2) there exists a pair of states p and q, respectively, in C_i and C_j such that p and q are TS-equivalent in C_{ij} (the reaching component from C_i to C_j). The state transition graph of an automaton is *TS-local* if and only if for

every SCC C_j of the graph, it is *TS*-local w.r.t. C_j. We give here a lemma which will be useful in the proofs of the next section

Lemma 2.6 (Kim et al. [9]). *Let C_1 and C_2 be SCCs such that C_1 is an ancestor of C_2 and C_2 is s-local. Let r and s be states, respectively, in C_1 and C_2 such that $\delta(r,x)=r$ and $\delta(s,x)=s$ for some $x\in\Sigma^+$. If there are states p and q, respectively, in C_1 and C_2, that are TS-equivalent in $C_{1,2}$, then r and s are also TS-equivalent in $C_{1,2}$.*

We can now state the characterization theorem du to Kim et al. [9].

Theorem 2.7 (Characterisation). *A minimal deterministic automaton is locally testable if and only if it satisfies the following conditions*:
1. *All SCCs of the state transition graph are s-local.*
2. *The state transition graph is TS-local.*

The polynomial time algorithm for testing if an automaton recognizes a locally testable language is based on the previous theorem. It is implemented in AGL, a set of Maple packages for manipulating automata, semigroups and languages. AGL is the last version of the software AG [6, 7].

2.3.3. Strongly locally testable languages

Strongly locally testable languages constitute a subclass of locally testable languages. Membership in a strongly locally testable language is determined only by factors of a fixed length k.

A *strongly k-testable language* L is a boolean combination languages such that, for $w\in\Sigma^+$, $I_k(w)$ the set of factors of length k of w, there exists a set X of words on Σ such that $w\in L$ if and only if $I_k(w)\subseteq X$. A language is *strongly locally testable* if it is strongly k-testable for some $k>0$.

Proposition 2.8. *Let L be a strongly k-testable language. Let u and v be two words such that u and v have the same factors of length k. Then we have*

$$u\in L\Leftrightarrow v\in L.$$

Strongly locally testable languages are often defined in an equivalent way as follows (see, e.g. [3]): L is strongly locally testable if it is a finite boolean combination of languages of the form $\Sigma^*w\Sigma^*$ where $w\in\Sigma^*$. Let us recall here an algebraic property of these languages which is given by Beauquier and Pin [3]. Denote by $S(L)$ the syntaxic semigroup of the recognizable language L on Σ^+ and by $\phi:\Sigma^+\to S(L)$ the syntaxic morphism of L. $P(L)=\phi(L)$ is the syntaxic image of L.

Theorem 2.9. *A language L is strongly locally testable if and only if its syntaxic semigroup $S(L)$ is locally idempotent and commutative and if $P(L)$ is a union of $\mathscr{D}$-classes of $S(L)$.*

3. Characterization of strictly locally testable languages

Local languages were studied in some detail in [14]. In [4] Berstel and Pin give definitions for local languages, which is exactly the definition of strictly 2-testable languages (see 2.2), and for local automata. They show that these automata recognize these languages.

Béal gives in [1] a more general definition for local automata. We make use of an equivalent definition in order to generalize the result stated by Berstel and Pin, that is to say, a strictly locally testable language is recognized by a local automaton.

Definition 3.1 (*Local automaton*). Let $\mathscr{M}=(\Sigma,Q,i,F,\delta)$ be a deterministic automaton and let k be a positive interger. The automaton $\mathscr{M}$ is k-local if for every w in Σ^k, the set $\{\delta(q,w)\,|\,q\in Q\}$ contains at most one element. An automaton is local if it is k-local for some $k>0$.

Proposition 3.2. *A language is strictly locally testable if and only if it is recognized by a trimmed minimal local automaton.*

Proof. Let $\mathscr{M}=(\Sigma,Q,i,F,\delta)$ be a k-local trimmed minimal automaton which recognizes L. Let us define the sets

$$X = \{w\,|\,\delta(i,w)=q\in Q \text{ and } |w|=k+1\},$$

$$Y = \{w\,|\,\delta(q,w)\in F \text{ and } |w|=k+1\},$$

$$Z = \{w\,|\,\delta(p,w)=q \text{ and } p,q\in Q, p\neq i, q\notin F, |w|=k+1\}.$$

Let $u=a_1\dots a_n$ be a word of L of length $\geqslant k+1$. The word u is the label of a successful path $c: i \xrightarrow{a_1} q_1\dots q_{n-1}\xrightarrow{a_n} q_n$. We have $a_1\dots a_{k+1}\in X$, $a_{n-k}\dots a_n\in Y$ and $a_j\dots a_{j+k}\in Z$, $\forall j,\ 1<j<n-k$. Consequently, a word of L satisfies the property of test sets of Definition 2.3.

Suppose now that $u=a_1\dots a_n$ is a word generated by these sets. Let us show that $i=q_0\xrightarrow{a_1}q_1\xrightarrow{a_2}\cdots\xrightarrow{a_n}q_n$ is a successful path. Seeing that $a_1\dots a_{k+1}\in X$, $\delta(i,a_1\dots a_{k+1})$ is defined and equal to q_{k+1}. Since $a_j\dots a_{j+k}\in Z$, $a_j\dots a_{j+k}$ is the label of a path $p\xrightarrow{a_j}\cdots q\xrightarrow{a_{j+k}} r$. Since $\mathscr{M}$ is k-local $\delta(p,a_j\dots a_{j+k-1})=\delta(q_{j-1},a_j\dots a_{j+k-1})$, so $q=q_{j+k-1}$ and then $r=q_{j+k}$. Finally, since $a_{n-k}\dots a_n\in Y$, $q_n\in F$, and then $i\to q_1\cdots\to q_n$ is a successful path in $\mathscr{M}$.

Conversely, suppose that L is strictly locally testable and suppose that $\mathscr{M}$, the trimmed minimal automaton recognizing L is not local. Then there exist two pairs of distinct states p,p' and q,q' and a word w of length k such that $\delta(p,w)=q$ and $\delta(p',w)=q'$. Let $u=a_1\dots a_n$ be a word of L such that w is a factor of u and such that p and q are on the path labeled u. Let $u'=a'_1\dots a'_n$ be a word of L such that w is a factor of u' and such that p' and q' are on the path labeled u'. Since L is strictly locally testable, all words having $a_1\dots a_l\cdot w$ or $a'_1\dots a'_{l'}\cdot w$ for prefix have exactly the same set of suffixes. It implies that $\delta(q,v)\in F\Leftrightarrow\delta(q',v)\in F$, and then q and q'

are equivalent (for the Nerode equivalence). Therefore the automaton is not minimal. Contradiction. □

Proposition 3.3. *An automaton $\mathscr{M}$ is local if and only if there do not exist two distinct states p and q and a word w such that $\delta(p,w)=p$ and $\delta(q,w)=q$.*

Proof. Suppose that $\mathscr{M}$ is not local. For every k, from a given rank, there exist a word $w=a_1\dots a_k$ of length k, and two sequences of states $p_0\dots p_k$ and $q_0\dots q_k$ such that $\delta(p_l,a_l)=p_{l+1}$ and $\delta(q_l,a_l)=q_{l+1}$. Let the rank be n^2 where n is the number of states of $\mathscr{M}$. We consider now the product automaton which states are pairs of states of $\mathscr{M}$ and where the transition function is defined as follows: $\delta'((r,r'),a)=(s,s')$ if $\delta(r,a)=s$ and $\delta(r',a)=s'$, for $a\in\Sigma$. Then consider the sequence of pairs $((p_l,q_l))_{0\leqslant l\leqslant n}$. Since $|w|>n^2$, there exists a pair of states which is matched twice on the path from (p_0,q_0) to (p_k,q_k). Let (p_i,q_j) be this pair. It implies that there exists a word u such that $\delta(p_i,u)=p_i$ and $\delta(q_j,u)=q_j$.

Conversely if there exist two distinct states p and q and a word u such that $\delta(p,u)=p$ and $\delta(q,u)=q$ then we have, for every $l\in\mathbb{N}^+$, $\delta(p,u^l)=p$ and $\delta(q,u^l)=q$. It implies that there exists an arbitrarily long word w such that $\{\delta(q,w)\mid q\in Q\}$ contains more than one element and hence $\mathscr{M}$ is not local. □

Proposition 3.4. *A language is strictly locally testable if and only if the state transition graph of its trimmed minimal automaton has the following properties*:
1. *All SCCs of the state transition graph are s-local.*
2. *All SCCs of the state transition graph are pairwise s-local.*

Proof. This proposition is a direct corollary of Proposition 3.3. Our polynomial time algorithm to decide whether a language is strictly locally testable is based on this property and is described in Section 5. □

4. Characterization of strongly locally testable languages

Strongly locally testable languages are characterized by a property of the strongly connected components of the state transition graph of their minimal automaton. This property leads us to a new polynomial time algorithm which is considered in the next section.

Definition 4.1. Let $\mathscr{M}=(\Sigma,Q,i,F,\delta)$ be an automaton and let C be an SCC of the state transition graph of $\mathscr{M}$. Let L_C^k be the languages relative to the strongly connected component C and to the positive integer k:

$$L_C^k=\{u\mid |u|>k,\exists x\in C \text{ such that } \delta(x,u)\in C\}.$$

The prefix language of the SCC C denoted by P_C is defined as follows:

$$P_C = \{u \mid \exists v \in \Sigma^* \text{ such that } \delta(i, u \cdot v) \in C\}.$$

Lemma 4.2. *Let L be a locally testable language and let $\mathcal{M} = (\Sigma, Q, i, F, \delta)$ be its minimal automaton. Let C be an SCC of the state transition graph of $\mathcal{M}$. If for every k from a given rank, $L_C^k \cap P_C \neq 0$ then there exists a word w of L_C^k and a state q in C such that $\delta(i, w) = \delta(q, w)$.*

Proof. Every word of L_C^k can be written $y \cdot u^l \cdot z$, $y, u, z \in \Sigma^*$, $l \in \mathbb{N}$, where there exists $r \in C$ such that $\delta(r, u) = r$. By assumption we can choose $y \cdot u^l \in P_C$. Let $p = \delta(i, y \cdot u^l)$. We can choose l large enough to have $\delta(p, u^l) = p$. There exists an SCC C' such that $p \in C'$. If $C = C'$ then, because the SCCs are s-local, we have $p = r$ and then there exists a state q such that $\delta(i, y \cdot u^l) = \delta(q, y \cdot u^l)$. Otherwise, if C and C' are dictinct SCCs, then C and C' are not pairwise s-local. Since the language is locally testable, the graph is TS-local. Because C' and C are successive SCCs, there exist two states $p_1 \in C'$ and $q_1 \in C$ such that for every word $v \in \Sigma^*$, $\delta(p_1, v)$ is in the reaching component from C' to C if and only if $\delta(q_1, v)$ is (p_1 and q_1 are TS-equivalent in the reaching component from C' to C). Now using Lemma 2.6, we have p and q are TS-equivalent in the reaching component from C' to C. Because u^l is in P_C there exists a word z such that $\delta(p, z) \in C$ and then we have $\delta(q, z) \in C$. Because of the TS-equivalence, there exists a word v such that $\delta(q, z \cdot v^{l'}) \in C$ and $\delta(p, z \cdot v^{l'}) \in C$ and so because C is s-local $\delta(q, z) = \delta(p, z)$ which lead us to conclude that there exists $w = y \cdot u^l \cdot z$ such that $\delta(i, w) = \delta(q, w)$. □

Proposition 4.3. *Let L be a locally testable language. Let $\mathcal{M} = (\Sigma, Q, i, F, \delta)$ be its minimal automaton and let S be its syntaxic semigroup. The two following properties are equivalent.*

1. *For every SCC C of the state transition graph of $\mathcal{M}$, there exists k such that $L_C^k \subseteq P_C$ or $L_C^k \cap P_C = \emptyset$.*
2. *$\forall x, y \in \Sigma^+$, $\phi(x) \mathscr{D} \phi(y) \Rightarrow \exists C$ an SCC of the state transition graph of $\mathcal{M}$ such that $\delta(i, x) \in C$ and $\delta(i, y) \in C$.*

In the following, we will write L_C for L_C^k whenever it is not ambiguous.

Proof. $1 \Rightarrow 2$.

Let x and y be two words such that $\phi(x) \mathscr{D} \phi(y)$ and $\delta(i, x) = q_x$, $\delta(i, y) = q_y$, $q_x \in C_1$, $q_y \in C_2$ and C_1 and C_2 are two distinct SCCs. It is obvious that two words having the same ϕ-image act the same on the states of $\mathcal{M}$. For $s \in S$ and $q \in Q$ we shall write $\delta(q, s)$ to designate the value $\delta(q, u)$ for any word u such that $\phi(u) = s$. There does not exist a pair of words u and v such that $\delta(q_x, u) = q_y$ and $\delta(q_y, v) = q_x$. It implies that there does not exist both elements of S, s and t such that $\phi(u) = s$ and $\phi(v) = t$ and such that $\phi(x) \cdot s = \phi(y)$ and $\phi(y) \cdot t = \phi(x)$, so we have $\neg(\phi(x) \mathscr{R} \phi(y))$. Since $\phi(x)$ and $\phi(y)$ are in the same $\mathscr{D}$-class, we have Fig. 1 as shown below.

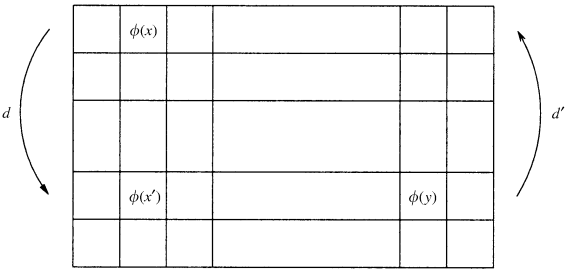

Fig. 1.

It follows that there exists $x' \in \Sigma^+$ such that $\phi(x)\mathscr{R}\phi(x')$ and $\phi(x')\mathscr{L}\phi(y)$. Let $q_{x'} = \delta(i,x')$. As $\phi(x)$ and $\phi(x')$ are in the same $\mathscr{R}$-class, there exists d and d' in S such that $\phi(x)\cdot d = \phi(x')$ and $\phi(x')\cdot d' = \phi(x)$. It comes that $\delta(q_x,d) = q_{x'}$ and $\delta(q_{x'},d') = q_x$ and so $q_{x'} \in C_1$. The fact that $\phi(x')$ and $\phi(y)$ are in the same $\mathscr{L}$-class is equivalent to say that there exist s and t in S such that $s\cdot\phi(x') = \phi(y)$ and $t\cdot\phi(y) = \phi(x')$. It implies $(t\cdot s)^k\cdot\phi(x') = \phi(x')$. Put u and v such that $\phi(u) = s$ and $\phi(v) = t$. Only the two following cases hold:

1. *Consider the case where* $\delta(i,(v\cdot u)^l) \in C_1$.
   Put $p = \delta(i,(v\cdot u)^l)$. By assumption we have $\delta(p,x') \in C_1$. Since $(v\cdot u)^{2l}\cdot x'$ has also the same ϕ-image as x', it follows that $\delta(p,(v\cdot u)^l\cdot x') \in C_1$. It implies that $(v\cdot u)^l \in L_{C_1}$; now $(v\cdot u)^l \in P_{C_1}$, therefore $L_{C_1} \cap P_{C_1} \neq \emptyset$.
   Suppose that $L_{C_1} \subseteq P_{C_1}$. We have $(v\cdot u)^l \in L_{C_1}$. As $\phi(x') = \phi((v\cdot u)^{l+1}\cdot x')$ and $\delta(p, (v\cdot u)^{l+1}\cdot x') \in C_1$, we have also $(v\cdot u)^{l+1} \in L_{C_1}$. Then we have $(v\cdot u)^l\cdot v \in L_{C_1}$ and $u\cdot(v\cdot u)^l \in L_{C_1}$. For every state $q \in C_1$, there exists a word $w \in \Sigma^*$ such that $\delta(q, w\cdot u\cdot(v\cdot u)^l\cdot x') = q_{x'}$. As $\phi(u\cdot(v\cdot u)^l\cdot x') = \phi(y)$ then we have $\delta(q, w\cdot y) = q_{x'}$. Thus $y \in L_{C_1}$. By assumption $y \in P_{C_1}$. Following the definition of P_{C_1}, y is such that there exists $z \in \Sigma^*$ and $\delta(i, y\cdot z) \in C_1$ which implies $\delta(q_y,z) \in C_1$ and so we have shown the existence of a path from q_y to $q_{x'}$. We show similarly the existence of a path from $q_{x'}$ to q_y, which contradicts the hypothesis. Thus we have $L_{C_1} \not\subseteq P_{C_1}$.
2. *Consider the case where* $\delta(i,(v\cdot u)^l) \notin C_1$.
   There exists an SCC C' such that $\delta(i,(v\cdot u)^{l+j}) \in C'$ for all $j \in \mathbb{N}$. Let us now define the two states $p = \delta(i,(v\cdot u)^l\cdot v)$ and $q = \delta(i,(u\cdot v)^l)$. It is obvious that $p \in C'$. Define C'' the SCC of q. We have $\delta(i,(u\cdot v)^l\cdot(u\cdot v)^l) \in C''$ and so $(u\cdot v)^l \in L_{C''} \cap P_{C''}$. We have also $(v\cdot u)^l\cdot v \in L_{C'} \cap P_{C'}$. As $(u\cdot v)^l \in L_{C'}$ and $(v\cdot u)^l\cdot v \in L_{C''}$, we have $L_{C''} \not\subseteq P_{C''}$ or $L_{C'} \not\subseteq P_{C'}$ or $C' = C''$. We now have to show that C' and C'' cannot be the same SCC. We have to distinguish the two following cases:
   (a) $p \neq q$.
   We show easily by the same proof as Proposition 3.3 that there exist two states p' and q' in the SCC C' and a word w such that $\delta(p',w) = p'$ and $\delta(q',w) = q'$ due to the fact that for all integer l', $\delta(p,(u\cdot v)^{l'}) \in C'$ and $\delta(q,(u\cdot v)^{l'}) \in C'$. It implies that $L(\mathscr{M})$ is not locally testable, contradicting the assumption.

(b) $p = q$.
We have $\delta(q, y) = q_y$ and $\delta(p, y) = \delta(i, (v \cdot u)^l \cdot v \cdot y) = \delta(i, (v \cdot u)^l \cdot x') = q_{x'}$.
It implies $q_{x'} = q_y$, contradicting the assumption.

In order to prove the converse part of the proposition, we will use a result stated by Beauquier and Pin [3]. Recall the definition of the congruence $\sim_k$. We have $u \sim_k v$ if:
1. u and v have the same prefix of length $< k$.
2. u and v have the same suffix of length $< k$.
3. u and v have the same factors of length k.

If L is locally testable then L is a union of $\sim_k$-classes for some k. Put $S_k = \Sigma^+ / \sim_k$. Then there exists a surjective morphism $\pi_k : S_k \to S$ and a subset Q of S such that

$$L = \pi_k^{-1}(Q) \quad \text{and} \quad Q = \pi_k(L),$$
$$Q = \pi^{-1}(P) \quad \text{and} \quad P = \pi(Q),$$

with $P = P(L)$ the syntaxic image of L.

Let $F_k(u)$ be the set of factors of length k of a word u. The following lemma is proved in [3].

Lemma 4.4. *Let u and v be two words of Σ^+. Then $\pi_k(u) \mathscr{D} \pi_k(v)$ if and only if $u = v$ or $F_k(u) = F_k(v) \neq \emptyset$.*

Now we close the proof of the converse part of Proposition 4.3. Let C be an SCC and w, w' be two words such that $w \in L_C \cap P_C$, $w' \in L_C$ and $w' \notin P_C$. Put $q_w = \delta(i, w)$. Because $w' \in L_C$, there exists a word z such that $\delta(i, w \cdot z \cdot w') \in C$. As $w \in L_C$, there exist a state q' and a word z' such that $\delta(i, w \cdot z \cdot w' \cdot z') = q'$ and $\delta(q', w) \in C$. Using Lemma 4.2, we can choose w such that $\delta(i, w) = \delta(q', w)$. Hence for all integers l, $\delta(i, (w \cdot z \cdot w' \cdot z')^l) \in C$. Because $\delta(i, w') \notin C$ we have $\delta(i, (w' \cdot z' \cdot w \cdot z)^l) \notin C$. Let $u = (w \cdot z \cdot w' \cdot z')^l$ and let $v = (w' \cdot z' \cdot w \cdot z)^l$. It is easy to notice that u and v have the same factors of length k for some k. It implies that $\pi_k(u) \mathscr{D} \pi_k(v)$ and then $\phi(u) \mathscr{D} \phi(v)$. In conclusion, if we have both $L_C \cap P_C \neq \emptyset$ and $L_C \not\subseteq P_C$ then there exist two words u and v such that $\phi(u) \mathscr{D} \phi(v)$ and u, v are not in the same SCC. □

Proposition 4.5. *Let L be a strongly locally testable language and let $\mathscr{M}$ be the minimal automaton recognizing L. Let x and y be two words on Σ^+. If $\phi(x) \mathscr{D} \phi(y)$ then there exists a strongly connected component C of $\mathscr{M}$ such that $\delta(i, x)$ and $\delta(i, y)$ are in C.*

Proof. As $\phi(x) \mathscr{D} \phi(y)$, there exist three words x', a, b such that $\phi(x \cdot a) = \phi(x')$ and $\phi(x') \mathscr{L} \phi(y)$. By Lemma 2.1, we have $\phi(x') \mathscr{D} \phi(y) \Rightarrow \pi_k(x') \mathscr{D} \pi_k(y)$. Since L is locally testable x' and y have the same factors of length k (Lemma 4.4). It is just the same for $x \cdot a \cdot (b \cdot a)^l$ and $y \cdot (b \cdot a)^l$. It follows that $u = x \cdot a \cdot (b \cdot a)^l \cdot z$ and $v = y \cdot (b \cdot a)^l \cdot z$ have the same factors of length k, $\forall z \in \Sigma^*$ (as far as l is great enough). Since L is strongly locally testable, $u \in L \Leftrightarrow v \in L$ (Proposition 2.8). Moreover, since $\mathscr{M}$ is minimal we

have $\delta(i, x\cdot a\cdot(b\cdot a)^l)=\delta(i, y\cdot(b\cdot a)^l)$, which implies $\delta(i,x')=\delta(i,y)$. Then $\delta(i,x)$ and $\delta(i,y)$ are in the same SCC. □

Theorem 4.6. *Let L be a language and let $\mathscr{M}=(\Sigma,Q,i,F,\delta)$ be its minimal automaton. L is strongly locally testable if and only if the following conditions are verified:*
1. *L is locally testable.*
2. *For every SCC C, $\forall p,q\in C$, $p\in F\Leftrightarrow q\in F$.*
3. *For every SCC C, there exists k such that $L_C^k\subseteq P_C$ or $L_C^k\cap P_C=\emptyset$.*

Proof. (1) and (3) imply that for all x,y belonging to the same $\mathscr{D}$-class, $q_x=\delta(i,x)$ and $q_y=\delta(i,y)$ belong to the same SCC (Proposition 4.3). Condition (2) tells us that q_x and q_y are both terminal states or both non-terminal states. It implies that $\mathscr{D}$-classes of S are saturated. So L is strongly locally testable.

Conversely (1) L is strongly locally testable implies that L is locally testable. (2) Suppose that there exist two states q_x and q_y of C such that $q_x\in F$ and $q_y\notin F$. There exist two words u and v such that $\delta(q_x,u)=q_y$ and $\delta(q_y,v)=q_x$. Let x be a word such that $\delta(i,x)=q_x$. The word $x\cdot(u\cdot v)^n$ is recognized and the word $x\cdot(u\cdot v)^n\cdot u$ is not recognized. Since these two words have the same factors of length k and since the language is strongly locally testable, we have $x\cdot(u\cdot v)^n\in L\Leftrightarrow x\cdot(u\cdot v)^n\cdot u\in L$. It leads us to a contradiction. (3) From Proposition 4.5, L being strongly locally testable, for two words x and y of the same $\mathscr{D}$-class, $\delta(i,x)$ and $\delta(i,y)$ are in the same SCC. By Proposition 4.3, it follows that $L_C\subseteq P_C$ or $L_C\cap P_C=\emptyset$ for every SCC C. □

5. Algorithms and complexity

5.1. Local automaton

In [9] we find the definition of the pair-graph of an automaton, as well as a lemma from which we deduce an algorithm to decide whether an automaton recognizes a strictly locally testable language.

Definition 5.1 (*Pair-graph*). Let $\mathscr{M}=(\Sigma,Q,i,F,\delta)$ be an automaton. Let Q_1 and Q_2 be two subsets of Q. Let $*$ be a symbol not in Q. The pair-graph on $Q_1\times Q_2$ is the edge-labeled directed graph $G(V,E)$, where $V=(Q_1\cup\{*\})\times(Q_2\cup\{*\})-\{(*,*)\}$ and E is defined as follows:

Define $\delta_i: Q_i\times\Sigma\to Q_i\cup\{*\}$, $i=1,2$, such that for all $p\in Q_i$ and $a\in\Sigma$,

$$\delta_i(p,a)=\begin{cases} q & \text{if } \delta(p,a)=q\in Q_i,\\ * & \text{if } \delta(p,a)\notin Q_i.\end{cases}$$

Then $E=\{((p,q),a,(r,s))\mid p\in Q_1, q\in Q_2, p\neq q,\ r\in Q_1\cup\{*\},\ s\in Q_2\cup\{*\},\ (r,s)\neq(*,*),\ \delta_1(p,a)=r,\ \text{and}\ \delta_2(q,a)=s\}$.

Notice that there is no out-going edge from a node $(p,*)$ or $(*,q)$ of the pair-graph and that a pair-graph is not necessarily connected.

Lemma 5.2. *Let C_i and C_j be two distinct SCCs of the state transition graph of an automaton $\mathscr{M}=(\Sigma,Q,i,F,\delta)$. Let Q_i and Q_j be, respectively, the set of states in C_i and C_j.*

1. *The component C_i is s-local if and only if the pair-graph on $Q_i \times Q_i$ has no cycle.*
2. *The components C_i and C_j are pairwise s-local if and only if the pair-graph on $Q_i \times Q_j$ has no cycle.*

Our algorithm proceeds from this lemma. Let G be the state transition graph of the automaton $\mathscr{M}=(\Sigma,Q,i,F,\delta)$ and let $n=|Q|$, $s=|\Sigma|$, m is the numbers of edges. For every strongly connected component C_i, let us call Q_i the set of its states. We can test the two conditions of Lemma 5.2 by running the following algorithm:

(1) *Find all SCCs by performing a classical algorithm* [2].
(2) *Let $|SCC|$ be the number of SCCs.*
(3) **for** *i* **from** 1 **to** $|SCC|$ **do**
(4) **for** *j* **from** *i* **to** $|SCC|$ **do**
(5) *Build the pair-graph $G_{i,j}$ on $Q_i \times Q_j$.*
(6) *Verify that it has no cycle.* **Lemma 5.2**
(* *otherwise exit, the language is not strictly locally testable* *)
(7) **endfor**
(8) **endfor**

Let us study the complexity of this algorithm. SCCs research (1) can be done in $O(\max(m,n)) \leqslant O(n^2)$. Suppose that G has r SCCs and that n_j is the number of states of the SCC C_j. Computation of (5) and (6) is achieved in $O(sn_in_j)$. Thus we can assert that the complexity of this algorithm is $O(\sum_{i=1}^{r}\sum_{j=i}^{r} sn_in_j) \leqslant O(sn^2)$.

5.2. Strongly locally testable automaton

We now introduce two new definitions and a theorem that lead us to an algorithm for testing whether an automaton recognizes a strongly locally testable language.

Definition 5.3 (*Product-graph of an SCC*). Let $\mathscr{M}=(\Sigma,Q,i,F,\delta)$ be an automaton. Let $C \subseteq Q$ be an SCC of $\mathscr{M}$. The product-graph of the SCC C is the directed graph $G(V,E)$ where $V=\{(p,q)=(\delta(i,w),\delta(r,w)) \mid r \in C, \delta(r,w) \in C, w \in \Sigma^*\}$ and $E=\{((p,q),a,(p',q')) \mid a \in \Sigma, \delta(p,a)=p', \delta(q,a)=q'\}$.

Definition 5.4 (*Attracting point*). Let $G=(X,U)$ be a directed graph. An attracting point is an SCC that has no descendant.

Theorem 5.5. *Let L be a locally testable language and let $\mathcal{M}=(\Sigma,Q,i,F,\delta)$ be its minimal complete automaton. The following two properties are equivalent*:

1. *For every SCC C of $\mathcal{M}$, the product-graph of C has exactly one attracting point.*
2. *For every SCC C of $\mathcal{M}$, there exists k such that $L_C^k \subseteq P_C$ or $L_C^k \cap P_C = \emptyset$.*

Proof. $1 \Rightarrow 2$.

By assumption, for every *SCC* C of $\mathcal{M}$, the product graph has only one attracting point. Suppose that property 2 is not verified. Then there exists an *SCC* C such that $L_C^k \cap P_C \neq \emptyset$ and $L_C^k \not\subseteq P_C$ for all k from a given rank. Using Lemma 4.2, there exists a word w of L_C^k and a state $q \in C$ such that $\delta(i,w)=\delta(q,w)$. Following the definition of the product-graph, we have $(\delta(i,w),\delta(q,w))$ is a vertex of the product-graph and then (p,p), $p \in C$ forms an SCC of the product-graph which is an attracting point. Let us now show the existence of a second attracting point. There exists a word w' such that $w' \in L_C$ and $w' \notin P_C$. Consider the pair (p_2,q_2) such that $p_2=\delta(i,w')$ and $q_2=\delta(r,w')$, with $r \in C$. The vertex (p_2,q_2) is in the product-graph. Let us notice that there exists no path from p_2 to p_1 because $w' \notin P_C$, and consequently, that there exists no path from (p_2,q_2) to the first attracting point. Then there exists necessarily a second attracting point and a path from (p_2,q_2) to this attracting point. Contradiction.

Conversely $2 \Rightarrow 1$. By assumption we have, for every SCC C, $L_C \cap P_C = \emptyset$ or $L_C \subseteq P_C$. Suppose we have an SCC C' such that the product-graph of C' has two attracting points. Let C_1 and C_2 be the two corresponding SCCs in $\mathcal{M}$.

1. Suppose $C_1 = C_2$. Then we have $\exists p, p' \in C_1$, $q \in C$, $w \in \Sigma^+$ with $p \neq p'$ such that $\delta(p,w)=p$, $\delta(p',w)=p'$ and $\delta(q,w) \in C$. It implies that the SCC C_1 is not s-local. Contradiction.
2. Suppose that $C_1 \neq C_2$. Then we have $L_C \subseteq L_{C_1}$, $L_C \subseteq L_{C_2}$; and there exist $w, w' \in L_C$ such that $\delta(i,w) \in C_1$, $\delta(i,w') \in C_2$. We have $w \in P_{C_1}$. Since $w' \in L_C$, we have $w' \in L_{C_2}$ and then $w' \in L_{C_2} \cap P_{C_2}$. So by assumption we have $L_{C_2} \subseteq P_{C_2}$. It implies that $\forall w \in L_C$, $w \in P_{C_1} \cap P_{C_2}$. We can conclude that the two SCCs C_1 and C_2 are successive ones. In the product-graph SCCs $C_1 \times C$ and $C_2 \times C$ lead to the same attracting point. Contradiction. □

Let G be the state transition graph of the automaton $\mathcal{M}=(\Sigma,Q,i,F,\delta)$ and let $n=|Q|$, $s=|\Sigma|$. For every strongly connected component C_j, let Q_j be the set of its states. We verify that the automaton recognizes a locally testable language with the algorithm described in [9]. Then we check the strong locality by using the following algorithm:

(1) *Find all the SCCs of the state transition graph.*
(2) **for** j **from** 1 **to** $|SCC|$ **do**
(3) *Build the product-graph of the SCC C_j.*
(4) *Verify that this graph has only one attracting point.*
(5) **endfor**

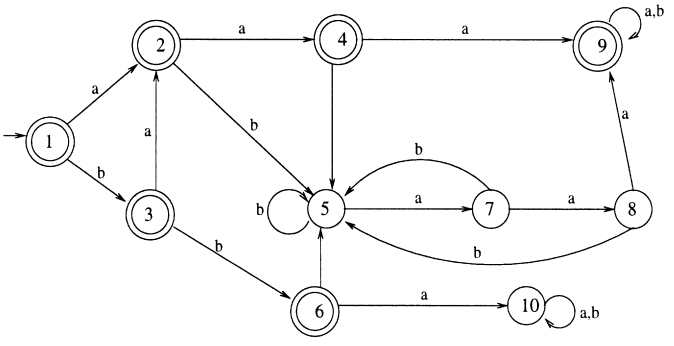

Fig. 2. Automaton $\mathscr{M}_1$.

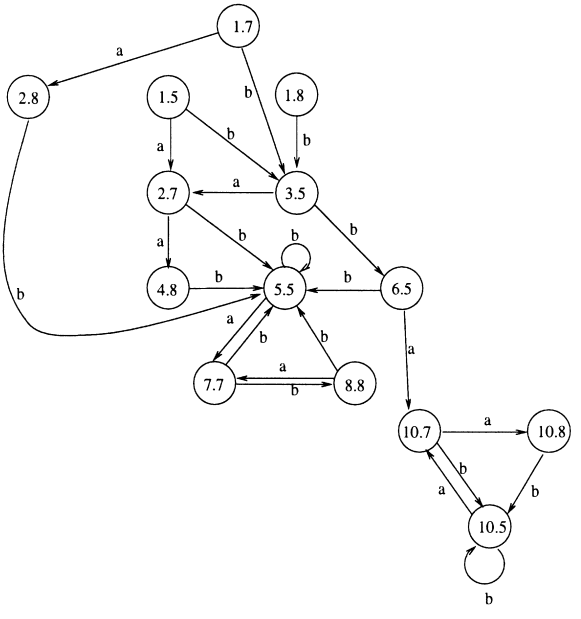

Fig. 3. Product-graph of the SCC $C=\{5,7,8\}$ of the automaton $\mathscr{M}_1$.

The complexity of local testability test is in $O(sn^2)$. The computation of steps (3) and (4) is achieved in $O(\sum_{i=1}^{r} s|Q|\times|Q_i|)\leqslant O(sn^2)$ where r is the number of SCCs.

Fig. 3 allows us to verify that $\mathscr{M}_1$ is not strongly locally testable. Indeed in the product-graph there are two attracting points $P_1=\{5.5,7.7,8.8\}$ and $P_2=\{10.7,10.8,10.5\}$.

The automaton $\mathscr{M}_2$ recognizes a strongly locally testable language. The product-graph of the SCC $C=\{3,4\}$ has only one attracting point $P_3=\{5.4,5.3\}$ (See Figs. 4, 5).

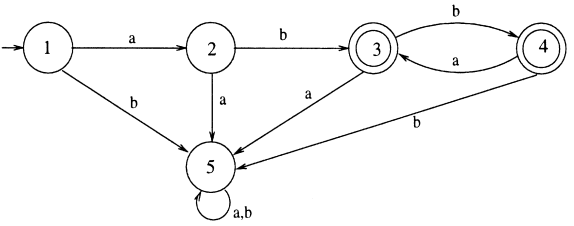

Fig. 4. Automaton $\mathcal{M}_2$.

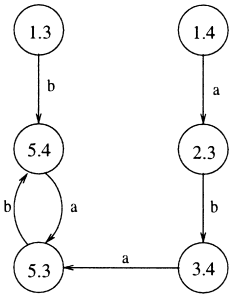

Fig. 5. Product-graph of the SCC $C = \{3, 4\}$ of the automaton $\mathcal{M}_2$.

In their paper [3], Beauquier and Pin introduce the notion of threshold locally testable languages. They give an algebraic characterization of this family of languages. The question is: is there an automaton characterization for these languages?

Acknowledgements

I thank the anonymous referees whose valuable suggestions improved the presentation of this paper.

References

[1] M.-P. Béal, Codes circulaires, automates locaux et entropie, Theoret. Comput. Sci. 57 (1988) 283–302.
[2] D. Beauquier, J. Berstel, P. Chrétienne, Éléments d'Algorithmique, Masson, Paris, 1992.
[3] D. Beauquier, J.-E. Pin, Scanners and languages, Theoret. Comput. Sci. 84 (1991) 3–21.
[4] J. Berstel, J.-E. Pin, Local languages and the Berry-Sethi algorithm, Theoret. Comput. Sci. 155 (1996) 439–446.
[5] J.A. Brzozowski, I. Simon, Characterization of locally testable events, Discrete Math. 4 (1973) 243–271.
[6] P. Caron, AG: a set of Maple packages for manipulating automata and finite semigroups, Software–Practice Experience 27 (8) (1997) 863–884.
[7] P. Caron, Langage: a maple package for automaton characterization of regular languages, in: D. Wood, S. Yu (Eds.), Automata Implementation: 2nd Internat. Workshop on Implementing Automata, WIA'97, Lecture Notes in Computer Science, vol. 1436, London, Ontario, Springer, Berlin, 1998, pp. 46–55.
[8] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley, Reading, MA, 1979.

[9] S.M. Kim, R. McNaughton, R. McCloskey, A polynomial time algorithm for the local testability problem of deterministic finite automata, IEEE Trans. Comput. 40 (10) (1991) 1087–1093.
[10] G. Lallement, Semigroups and Combinatorial Applications, Wiley-Interscience, New York, 1979.
[11] A. De Luca, A. Restivo, A characterization of strictly locally testable languages and its application to subsemigroups of a free semigroup. Inform. and Control 44 (1980) 300–319.
[12] R. McNaughton, Algebraic decision procedures for local testability, Math. Systems Theory 8 (1974) 60–76.
[13] R. McNaughton, S. Papert, Counter Free Automata, MIT Press, Cambridge, 1971.
[14] M. Nivat, Transduction des langages de chomsky, Ann. Inst. Fourrier Grenoble 18 (1) (1968) 339–456.
[15] D. Perrin, Finite automata, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Formal Models and Semantics, vol. B, Elsevier, Amsterdam, 1990, pp. 1–57.
[16] J.-E. Pin, Variétés de langages et variétés de semigroupes, Thèse d'état, Université Paris 6, France, 1981.
[17] J.-E. Pin, Variétés de Langages Formels, Masson, Paris, 1984.
[18] M.-P. Schützenberger, Sur certaines opérations de fermeture dans les langages rationnels, Symp. Math. 15 (1975) 245–253.
[19] Y. Zalcstein, Locally testable languages, J. Comput. System Sci. 6 (1972) 151–167.