A Procedure for K-Local Testability

Antonio Magnaghi, Hidehiko Tanaka

ローカルテスタビィティ ＬＴ（Ｌｏｃａｌ Ｔｅｓｔａｂｉｌｉｔｙ）は、正式言語の分野におけるリサーチ・アクティブ・エリアを表すものである。 ＬＴはパターン認識の研究の根幹をなすものである。スプリングスのこのような特性を理論上とストリングスの並行分解の観点に対応するところからまた、エラー探知上から考察したものである。これは、狭義におけるＫ－ｌｏｃａｌｌｙ－ｔｅｓｔａｂｌｅ－言語の研究と、ＬＴｋｓ．ｓ．の特性における決定テクニックの可能性について焦点をあわせたものである。この研究の結果により、直接的にＬＴｋｓ．ｓにシンクロタッチ・モノイドによる複雑な代数的アポローチを採用する事なく、ストリングス言語が入っているかどうか確かめる事が可能になる。さらに我々の方法はＬＴｋｓ．ｓ．言語の順序の決定問題と密につながっており、ｋ－ｌｏｃａｌｌｙ－ｔｅｓｔａｂｌｅのような言語パラメーターｋ（最小の数値）まで定義するする事が可能になる。

A Procedure for K-Local Testability

Antonio Magnaghi, Hidehiko Tanaka
Graduate School of Electrical Engineering
The University of Tokyo
7-3-1 Bunkyo-ku, Tokyo 113, Japan
e-mail: {magnaghi,tanaka}@mtl.t.u-tokyo.ac.jp

The concept of local testability (LT) represents an active area of research in the field of formal languages. LT has its roots in the study of pattern recognition. Interest in such a property springs both from theoretical and applied concerns about parallel parsing of strings and error detection. This paper focuses on the class of k-locally-testable languages in a strict sense ($LT_k s.s.$) and on a possible decidibility technique for $LT_k s.s.$ property. We prove a decidibility algorithm, basing directly on specific structural properties of $LT_k s.s$. Such a result enables to ascertain whether a string language is in $LT_k s.s.$ in a direct manner, without adopting the elaborated algebraic approach of the syntactic monoid. In addition, our method strictly relates to the problem of determining the order of a $LT_k s.s.$ language, being the order defined as the smallest value of the parameter k such that the language is k-locally-testable.

1. Introduction

The concept of local testability (LT) has been broadly investigated in the previous decades, and it still represents an active area of research in the field of formal languages.
LT has its roots in the study of pattern recognition. It is possible to identify two main research threads. One of them is concerned with linear sequences of symbols, i.e. string languages, whereas the other analyzes more articulated structures, such as, for instance, images and tree languages[7),10),11)]. In the case of strings the wider class of Aperiodic Languages constitutes the formal framework for LT[2),8)]. Aperiodicity reveals to be a linguistic universal, characterized in a variety of ways: grammatical inference[4)], neural networks[8)] and algebraic structures[2),3),8),12)]. The importance of LT springs from its close link to aperiodicity. Requiring locally testable class (LT) to be closed w.r.t. concatenation leads to aperiodic languages. A hierarchy of aperiodic languages was identified[2)] imposing different constrains on the recognition process of strings. It comprises Definite, Reverse Definite, Locally Testable in a Strict Sense, Locally Testable and properly Aperiodic or Non-Counting languages at the top of the taxonomy.

This paper focuses on the class of k-locally testable languages in a strict sense ($LT_k s.s.$) and on a possible decidibility algorithm for $LT_k s.s.$ property. Interest in local testability in a strict sense relies on both theoretical and applied concerns about parallel parsing of strings and error detection.

A systematic characterization of the different sub families of aperiodic languages was presented in the past[1),3)] and decidibility algorithms for them were formulated[1),5)]. However, the result was quite difficult and moreover for the specific case of Local Testability in a strict sense ($LT s.s.$) the decidibility problem is solved in only an indirect way, through a modification of a more general and complex procedure for LT. This paper reports on our efforts to prove a different decidibility technique, relaying directly on specific structural properties of $LT_k s.s.$ A necessary and sufficient condition is formulated for $LT_k s.s.$ By these means it is possible to ascertain whether a string language is in $LT_k s.s.$ in a direct manner, without adopting the elaborated algebraic approach of the syntactic monoid.

The proposed algorithm employs convenient sets of k-length strings associated to the nodes of an accepting automaton for the language L, and through the analysis of their set-theoretical characteristics it clarifies those properties required to guarantee $LT_k s.s.$ This approach presents some worthwhile advantages. In addition to its simplicity, it allows to obtain other interesting results about $LT_k s.s.$ in a straightforward way, whereas without it some more considerations would be needed. Our method relates directly to the problem of determining the order of a $LT_k s.s.$ language, being the order defined as the smallest value of the parameter k such that the language is k-locally testable. The order of an automaton, equivalently of the accepted language, clearly emerges to be dependent on intersection properties of paths that start at different nodes and are labeled through the same string of symbols. Hence a topological aspect is directly related to the set of strings employed for the syntactic analysis of a testable sentence.

The extension of these considerations to tree languages is a relevant aspect to investigate. Generally the difficulties encountered in such an effort are also due to the impossibility of an immediate extension of the conceptual and formal tools developed in the string case. As the proposed decidibility algorithm does not utilize the syntactic monoid approach and its algebraic properties, it might give some insights on how to adapt our considerations to tree languages.

2. Basic definitions

LT has its roots in the study of pattern recognition. Intuitively let x be a string composed by concatenating letters from an alphabet. The recognition procedure is carried out on x through a window of fixed arbitrary length to be moved along the string. The sequences of symbols observed through the window are annotated in a record, regardless of the order in which they are met and of the position they occupy in the string. After moving the window from one end to the other, x is accepted or rejected basing on the set of substrings that compose the produced record. Local testing proved to be a general concept, able to capture many situations.

The recognition procedure described above can be modified and generalized to a variety of contexts: Definite, Reverse-Definite, Generalized-

Definite, Locally Testable in a strict sense ($LTs.s.$), Locally-Testable (LT) and Aperiodic languages. In this section, basing mainly on the work of McNaughton and Papert[8)], we will present the definition of k-Locally Testability in a strict sense ($LT_k s.s.$) and some of its properties essential to the following considerations.

Let Σ be a finite alphabet of symbols, and let Σ^* denote the free monoid over Σ, including all the strings obtained by concatenation of alphabet elements. A subset L of Σ^* is a string language, or a string event, over Σ. If L defines a regular set, i.e. it can be characterized through a regular expression, the language L is a regular language and it can be recognized by a finite state automaton M.

Being k a non-negative integer number, it is possible to define the following operators on a string x of length greater or equal to k:

$$x = a_1 a_2 \dots a_n (a_i \in \Sigma, 1 \leq i \leq n, n \geq k)$$
$$L_k(x) = \{y : x = yw \wedge |y| = k\} \quad (1)$$

$$R_k(x) = \{w : x = yw \wedge |w| = k\} \quad (2)$$

$$I_k(x) = \{w : x = ywz \wedge y, w, z \neq \epsilon \wedge |w| = k\} (3)$$

The operator $L_k(x)$ extracts the k-length prefix from input string. Symmetrically, $R_k(x)$ produces the k-length suffix of word x. Equation (3) defines the set of properly internal k-length sub strings of x. If the length of x (denoted by $|x|$) equals k or (k+1), $I_k(x)$ is the empty set.

Let α_k,β_k,γ_k be sub sets of Σ^k: they are sets of strings over Σ whose length is k.

The language L is k-locally testable in a strict sense ($L \in LT_k s.s.$) if sets α_k,β_k,γ_k exist such that for every $x \in \Sigma^*$ ($|x| \geq k$):

$$(x \in L) \Longleftrightarrow$$
$$(L_k(x) \in \alpha_k \wedge I_k(x) \subseteq \beta_k \wedge R_k(x) \in \gamma_k) \quad (4)$$

Basing on relation (4), a k-locally testable language in a strict sense has the property that syntactic analysis can be performed locally. On a procedural level, parsing activity requires to extract from string x its prefix ($L_k(x)$), suffix ($R_k(x)$) and the set of internal sub strings ($I_k(x)$). Recalling the initial window analogy, x can be parsed by a k-letters-wide loophole to be moved from left to right end one symbol at a time. Correctness is evaluated using only the information collected through such a decomposition. In particular $\alpha_k, \beta_k, \gamma_k$ contain the recognition patterns to utilize in order to ascertain whether the string belongs to L or not. α_k can be interpreted as the set containing all possible k-length prefixes of strings of L. Dually, γ_k is the set of all acceptable k-length suffixes. β_k is the set of all acceptable internal k-length sub strings of words of L. It should be noted that no information about order or relative position of occurrence is kept.

Locality property shows a link to parallel parsing of string languages. A word of a local language has such a syntactical structure that allows to analyze each sub string independently from all the others. Hence, it is possible to decompose the input sentence among computational units of a parallel computer to simultaneously recognize the different parts, substantially improving parsing performance. Moreover, another feature emerges for $LTs.s.$ languages in relation to error identification. The presence of a syntax error is easy to detect and its position is precisely defined as well: it is located within the k-length sub string that does not match any element of $\alpha_k, \beta_k, \gamma_k$. On the contrary, in the general case when $LTs.s.$ property does not hold, error handling is more complex.

Definition (4) does not consider strings of L consisting of a number of symbols less than k. In this case the number of possible words is limited, so parsing can be performed separately in a simple way.

As an example of $LT_k s.s.$ language, let us consider the set of words over $\Sigma = \{a, b\}$ such that every possible occurrence of letter a is immediately followed by the string bb: $abbb$, bbb are correct words; $ababb$, ba are not. The defined language is in $LT_k s.s.$ for k=3,4,5,... If we define, over the same alphabet, the language whose words contain an even number of occurrences of symbol a, local testability property never holds for any value of k, because the involved constraint is not local.

The remainder of this paper will focus on the problem of deciding whether a language L is in $LT_k s.s.$ or not. We will assume L to be defined through a deterministic finite state automaton (DFA) M. After establishing the decidibility result for $LT_k s.s.$ property, we will also be able to obtain a procedure to determine the order of the language.

Before concluding this section, a theorem is stated expressing an upper bound to parameter k value for $LT_k s.s.$ property. It is a direct consequence of what was proved by Brzozowski and Simon[1], adapted to our framework. Such a result will allow us to show in the next section how $LT_k s.s.$ decidibility leads to an algorithm for $LT s.s.$ decidibility and for language order identification.

Theorem1 Let M be a DFA accepting a language L in $LT s.s.$, and let $k = n+1$, being n the number of states of M. Then M is $LT_{k+1} s.s.$

3. Notations

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA accepting L: Q is the set of states, Σ the input alphabet, δ the transition function, $q_0 \in Q$ the initial state and $F \subseteq Q$ a non empty set of final states. For any $q \in Q$ and $x \in \Sigma^*$, $\delta(q, x)$ denotes the resulting state when input string x is applied to M in state q.

Let $G = (V, \Sigma, P, < q_0 >)$ be the linear right-derivative context-free grammar associated univocally to M as follows:

(1) the set V of non terminals contains a symbol for every state of M:
$$V = \{< q >: q \in Q\}$$

(2) the terminal alphabet Σ of G equals the input alphabet Σ of M

(3) the set P of linear productions of G is:
$$P = \{< q_1 > \rightarrow a < q_2 >: q_1, q_2 \in Q \wedge \wedge a \in \Sigma \wedge \delta(q_1, a) = q_2\} \cup \cup \{< q_f > \rightarrow \epsilon : q_f \in F\}$$

(4) the axiom of the grammar $< q_0 >$ corresponds to the initial state q_0 of M.

Let π be the omomorphism whose domain and image are respectively $(\Sigma \cup V)^*$, Σ^*:

$$\pi(a) = a, \forall a \in (\Sigma \cup \{\epsilon\})$$
$$\pi(< q_j >) = \epsilon, \forall q_j \in Q$$
$$\pi(xy) = \pi(x)\pi(y), \forall x, y \in (\Sigma \cup V)^*$$

For any state q_i of M and for any integer k, the following set of strings is defined:

$$V_k(q_i) = \{\pi(\omega) :< q_i > \stackrel{k}{\Longrightarrow} \omega\} \qquad (5)$$

$V_k(q_i)$ is the set of all words obtained through the application of $\pi(.)$ to derivations of length k starting from the non terminal $< q_i >$. If the last derivation to produce ω is not terminal $|\pi(\omega)| = k$, otherwise $|\pi(\omega)| = (k-1)$.

For every $q_i \in Q$ and for every $x \in V_k(q_i)$, let us define the set $D_k(q_i, x)$:

$$D_k(q_i, x) = \{y = xu \in V_{k+1}(q_i) : u \in \Sigma \cup \{\epsilon\}\}, if|x| = k \qquad (6)$$

$$D_k(q_i, x) = \emptyset, if|x| = (k-1) \qquad (7)$$

$D_k(q_i, x)$ consists of the strings whose k-prefix equals x and that are produced by a chain of (k+1) derivations from $< q_i >$.

Lemma1 $x \in D_k(q_i, x)$ iff $\delta(q_i, x) \in F$

Proof

x belongs to $V_k(q_i)$, hence its length can be either (k-1) or k.

A generic string in $D_k(q_i, x)$ consists necessary of k or (k+1) symbols. By hypothesis, $x \in D_k(q_i, x)$, therefore $|x| = k$. In definition (6), the symbol u must equal ϵ: thus $x \in V_{k+1}(q_i)$. The unique way to obtain a string of k letters by (k+1) derivations in G is to require that the last derivation is terminal: $< q_i > \stackrel{k}{\Longrightarrow} x < q_j > \stackrel{1}{\Longrightarrow} x$, where $< q_j > \rightarrow \epsilon$. If the rule $< q_j > \rightarrow \epsilon$ is included in the set of productions of G, then $q_j \in F : \delta(q_i, x) = q_j \in F$.

□

4. Decidibility procedure for $LT_k s.s.$

In this section we will propose a possible algorithm to decide if a regular language L is in $LT_k s.s.$ for a fixed value of k. The adopted approach will allow us to answer (with "yes" or "not") to the question: *"Is L k-locally testable in a strict sense in the case k equals a specific number* $\tilde{k}$*?"*. Such a formulation corresponds to the decision-problem derived from the optimization-problem about the order of L.

We assume that the language L is given through an automaton accepting it. A regular language can generally be recognized by different automata: only its minimal accepting automaton is unique, neglecting isomorphisms that rename its states. An interesting aspect will emerge from the remainder. Our procedure works correctly regardless of the particular representation of the accepting automaton M, allowing us to speak of a $LT_k s.s.$ language L or equivalently of $LT_k s.s.$ property of an arbitrary automaton accepting L. In particular we will not require M to be reduced, whereas many results available in literature focus on the minimal automaton accepting L.

More precisely, let the language L be defined through a DFA, that does not contain unreachable states from the initial node q_0 or states from which it is not possible to reach a final state $q_f \in F$. Under these assumptions, the transition function δ will generally be a partial function over $Q \times \Sigma$.

Before proceeding, the following sets are introduced for language L:

$$\alpha_k = \{x : w = xy \wedge |x| = k \wedge w \in L\} \quad (8)$$
$$\beta_k = \{v : w = uvz \wedge u, z \neq \epsilon \wedge |v| = k \wedge w \in L\} \quad (9)$$
$$\gamma_k = \{y : w = xy \wedge |y| = k \wedge w \in L\} \quad (10)$$

It is clear that, if $L \in LT_k s.s.$, through $\alpha_k, \beta_k, \gamma_k$ syntactical analysis can be correctly carried out. On the contrary, if $L \notin LT_k s.s.$, the language recognized through such sets is a super set of L.

Theorem2 Let L be $LT_k s.s.$, then every DFA M accepting L is such that:

$D_{k-1}(q_i, x) = D_{k-1}(q_h, x)$

for any $q_i, q_h \in Q$, for any $x \in V_{k-1}(q_i) \cap V_{k-1}(q_h)$.

Proof

For convenience reasons, a new non final state P ($\notin Q$) is introduced in order to let δ become a total function over $(Q \cup \{P\}) \times \Sigma$. From a state $q_i \in Q$ now transitions are defined in correspondence of every $a \in \Sigma$: $\delta(q_i, a) = P$ if previously there was no edge leaving q_i with label a. P is an adsorbing state in the sense that $\delta(P, a) = P$ for any $a \in \Sigma$, and being P not final, if it is reached, it means that the input x contains a syntactic error, causing the rejection of the string. The introduction of P does not alter the generality of the considerations below.

It will be proved that $L \notin LT_k s.s.$ if there exist two distinct states q_i, q_h and a string x such that:

$$x \in V_{k-1}(q_i) \cap V_{k-1}(q_h) \quad (11)$$
$$D_{k-1}(q_i, x) \neq D_{k-1}(q_h, x) \quad (12)$$

If $|x| = (k-2)$, then $D_{k-1}(q_i, x) = D_{k-1}(q_h, x) = \emptyset$ because of definition (7). Hence, necessarily $|x| = k-1$.

Let x have the form: $x = t_1 t_2 \dots t_{k-1}, 1 \leq i \leq k-1, t_i \in \Sigma$. Condition (12) implies that $q_i \neq q_h$ and that at least one of the sets $(D_{k-1}(q_i, x) - D_{k-1}(q_h, x)), (D_{k-1}(q_h, x) - D_{k-1}(q_i, x))$ is not empty. For instance, let y belong to $D_{k-1}(q_i, x) - D_{k-1}(q_h, x) : y = xt_k(|y| = k)$. As $y \notin D_{k-1}(q_h, x)$, $\delta(q_h, y) = P$. Being M deterministic, and $q_i \neq q_h$, there must be two different strings w_1, w_2 that lead from q_0 to q_i and q_h respectively: $\delta(q_0, w_1) = \delta(q_0, a_1 a_2 \dots a_m) = q_i; \delta(q_0, w_2) = \delta(q_0, b_1 b_2 \dots b_n) = q_h$, where $m, n \geq 0$, but not both of them can equal zero, and $w_1 \neq w_2$. Let us consider the following states: $\tilde{q_i} = \delta(q_i, x)$; $\tilde{q_h} = \delta(q_h, x)$; $\tilde{q_i}' = \delta(\tilde{q_i}, t_k), P = \delta(\tilde{q_h}, t_k)$. Basing on the characteristic of M, that is without useless states when evaluating $D_{k-1}(q_i, x)$ and $D_{k-1}(q_h, x)$ (P is not added in that context), from $\tilde{q_i}'$ a final state q_f is reachable through the string $z = c_1 c_2 \dots c_s, (s \geq 0) : \delta(\tilde{q_i}, z) = q_f \in F$. Now, let us consider the word $w = w_2 yz$, where the length of w is greater or equal to k. $\delta(q_0, w_2 y) = P$ implies $\delta(q_0, w_2 yz) = \delta(q_0, w) = P$, hence $w \notin L$.

Let us consider $L_k(w)$, two cases are possible: (a)if $n \geq k$, $L_k(w) = b_1 b_2 \dots b_k$, (b)if $0 \leq n < k$, $L_k(w) = b_1 b_2 \dots b_n t_1 t_2 \dots t_l$, where $n + l = k, 1 \leq l \leq k$.

(Case a) $\delta(q_0, w_2) = q_h$, and from q_h a final state $q_{f'}$ is reachable, hence from $\delta(q_0, b_1, b_2 \dots b_k)$ the same node $q_{f'}$ is reachable. As a consequence, $b_1 b_2 \dots b_k$ is the prefix of a string in L, hence because of (8) $L_k(w) \in \alpha_k(n \geq k)$.

(Case b)The condition $x = t_1 t_2 \dots t_k \in V_{k-1}(q_h)$ guarantees that for any state: $q_l = \delta(q_0, w_2 t_1 t_2 \dots t_l), 1 \leq l \leq k$, a path exists leading to a final state of M from q_l. Hence, it is possible to conclude again that $L_k(w) \in \alpha_k(0 \leq n \leq k)$.

Let us consider $R_k(w)$. (a)if $s \geq k$, $R_k(w) = c_{s+1-k} c_{s+2-k} \dots c_s$; (b)if $0 \leq s < k$, $R_k(w) = t_l t_{l+1} \dots t_k c_1 c_2 \dots c_s, 1 \leq l \leq k$. Both in case (a) and (b) we can prove that $R_k(w) \in \gamma_k$ in a fashion similar to the one used for $L_k(w)$.

Finally it is also possible to verify that $I_k(w) \subseteq \beta_k$.

Hence if conditions (11), (12) simultaneously hold, a string w exists such that $x \notin L$, but for which: $L_k(w) \in \alpha_k, I_k(w) \subseteq \beta_k, R_k(w) \in \gamma_k$. Thus we conclude $L \notin LT_k s.s.$

□

In order to prove theorem 2, a preliminary lemma is required.

Lemma2 Let M be a DFA such that:

$D_{k-1}(q_i, x) = D_{k-1}(q_h, x)$

for any $q_i, q_h \in Q$ and for any $x \in V_{k-1}(q_i) \cap V_{k-1}(q_h)$. Then the state $\delta(q_i, x)$ is equivalent to $\delta(q_h, x)$.

Proof

Also in this case, it can be convenient to introduce the state P (see proof of theorem 2), in order not to distinguish tedious particular cases in the considerations below.

Let $x = a_1a_2 \ldots a_k \in V_{k-1}(q_i) \cap V_{k-1}(q_h)$, and let us assume that $\tilde{q_i} = \delta(q_i, x)$ is not equivalent to $\tilde{q_h} = \delta(q_h, x)$. This implies the existence of a string $y = b_1b_2 \ldots b_m (m \geq 0)$ such that $\delta(\tilde{q_i}, y) \in F$ and $\delta(\tilde{q_h}, y) \notin F$ or $\delta(\tilde{q_i}, y) \notin F$ and $\delta(\tilde{q_h}, y) \in F$. Considering for instance the first possible case, let $z = xy(|z| \geq k)$, and π_i (resp. π_h) be the path comprising the edges of M used by the involved transitions from q_i (resp.q_h) to $\delta(q_i, z)$ (resp. $\delta(q_h, z)$):

$$\pi_i = (q_i, \delta(q_i, a_1))(\delta(q_i, a_1), \delta(\delta(q_i, a_1), a_2)) \ldots (\delta(q_i, a_1a_2 \ldots a_kb_1b_2 \ldots b_{m-1}), b_m)$$

$$(resp. \pi_h = (q_h, \delta(q_h, a_1))(\delta(q_h, a_1), \delta(\delta(q_h, a_1), a_2)) \ldots (\delta(q_h, a_1a_2 \ldots a_kb_1b_2 \ldots b_{m-1}), b_m))$$

With $\tilde{q_i\prime}$ (resp. $\tilde{q_h\prime}$) we designate the node at a distance of (k-1) edges from $\delta(q_i, z)$ (resp. $\delta(q_h, z)$) along the path π_i (resp. π_h). As $|z| \geq k$, $\tilde{q_i\prime}$ (resp. $\tilde{q_h\prime}$) exists, and $\tilde{x}$ is the (k-1)-suffix of z such that: $\delta(\tilde{q_i\prime}, \tilde{x}) = \delta(\tilde{q_i}, y)$ (resp. $\delta(\tilde{q_h\prime}, \tilde{x}) = \delta(\tilde{q_h}, y)$). We notice that $\tilde{x} \in V_{k-1}(\tilde{q_i\prime}) \cap V_{k-1}(\tilde{q_h\prime})$, however the set equality $D_{k-1}(\tilde{q_i\prime}, \tilde{x}) = D_{k-1}(\tilde{q_h\prime}, \tilde{x})$ does not hold. $\delta(\tilde{q_i\prime}, \tilde{x}) = \delta(\tilde{q_i}, y) \in F$, hence lemma 1 guarantees that $\tilde{x} \in D_{k-1}(\tilde{q_i\prime}, \tilde{x})$. On the other hand, being $\delta(\tilde{q_h\prime}, \tilde{x}) = \delta(\tilde{q_h}, y) \notin F, \tilde{x} \notin D_{k-1}(\tilde{q_h\prime}, \tilde{x})$ (lemma 1). This represents a contradiction: necessarily $\delta(q_i, x)$ is equivalent to $\delta(q_h, x)$.
□

We can derive the following result as immediate consequence of previous lemma:

Corollary1 Let M be a reduced DFA such that:

$$D_{k-1}(q_i, x) = D_{k-1}(q_h, x)$$

for any $q_i, q_h \in Q$ and for any $x \in V_{k-1}(q_i) \cap V_{k-1}(q_h)$. Then $\delta(q_i, x)$ equals $\delta(q_h, x)$

It is now possible to proceed to theorem 3: it proves the validity of exchanging hypothesis and thesis in theorem 2.

Theorem3 Let M be a DFA such that:

$$D_{k-1}(q_i, x) = D_{k-1}(q_h, x)$$

for any $q_i, q_h \in Q$ and for any $x \in V_{k-1}(q_i) \cap V_{k-1}(q_h)$. Then the language accepted by M is $LT_k s.s.$ w.r.t. $\alpha_k, \beta_k, \gamma_k$.

Proof

Recalling the definition of $LT_k s.s.$ language (4), two implications must be verified, being $w \in \Sigma^*, |w| \geq k$:

$$w \in L \Rightarrow L_k(w) \in \alpha_k \wedge I_k(w) \subseteq \beta_k \wedge R_k(w) \in \gamma_k$$

$$L_k(w) \in \alpha_k \wedge I_k(w) \subseteq \beta_k \wedge R_k(w) \in \gamma_k \Rightarrow w \in L$$

Because of (8), (9), (10) the first of them holds in a straightforward manner, whereas the second one requires additional considerations.

Let w be a string of this form: $w = a_1a_2 \ldots a_m (m \geq k)$, with the property that $L_k(w) \in \alpha_k, I_k(w) \subseteq \beta_k, R_k(w) \in \gamma_k$. Our aim is to show that w is syntactically correct. Let q_r be the state of M reached from q_0 through $a_1a_2 \ldots a_r$, r-prefix of w: $q_r = \delta(q_0, a_1a_2 \ldots a_r)(1 \leq r \leq m)$. In the general case δ is not a total function over $Q \times \Sigma$, and w is a sentence of L iff the following conditions hold:

- transition
  $$\delta(q_r, a_{r+1}) \qquad (13)$$
  is defined for any $0 \leq r \leq m-1$,
- and
  $$q_m \in F \qquad (14)$$

As a first step, condition (13) will be proved by complete induction on r.

Extending the terminology from one letter input to a i-letters string ($i > 1$), we will say that $\delta(q_r, a_{r+1}a_{r+2} \ldots a_{r+i-1})$ is defined if all the following transitions are defined: $\delta(q_{r+j}, a_{r+1+j})$,$0 \leq j \leq i-2$.

(Base of induction: r=0)

$L_k(w) = a_1a_2 \ldots a_k \in \alpha_k$. Recalling the definition of α_k (8), a string y exists such that y is in L and its k-prefix is $a_1a_2 \ldots a_k$, hence $\delta(q_0, a_1)$ is defined, being y correct.

(Inductive assumption)

Let us assume that $\delta(q_j, a_{j+1})$ is defined for $1 \leq j \leq r-1$, where $1 \leq r \leq m-1$.

(Inductive step)

Now it is required to show that the inductive assumption holds also for $j = r$. Two different cases are considered.

(Case a: $1 \leq r \leq k-1$) $L_k(w) = a_1a_2 \ldots a_k \in \alpha_k$. Basing on the same considerations expressed previously (Base of induction), it is possible to conclude that $\delta(q_j, a_{j+1})$ is defined for any $1 \leq j \leq k-1$.

(Case b: $k \leq r \leq m-1$) Let u be the string of (k-1) symbols: $a_{r-k+2}a_{r-k+3} \ldots a_r$, and let v be ua_{r+1}. v is a k-length sub string of the input string w. It belongs to $(\beta_k \cup \gamma_k)$, its initial character a_{r-k+2} is at least the second one: it occurs in position number (r-k+2) from left, which is greater or equal to 2. Hence a string y in L exists such that $v \in I_k(y)$ or $v = R_k(y)$. In both cases, we can consider two states $\tilde{q}$ and $\tilde{q'}$ such that: $u \in V_{k-1}(\tilde{q}), \tilde{q'} = \delta(\tilde{q}, u)$. From $\tilde{q'}$ the output transition $\delta(\tilde{q'}, a_{r+1})$ is necessarily defined because all the transitions in the sequence $\delta(\tilde{q}, v)$ are defined ($v \in I_k(y)$ or $v = R_k(y)$) and $\delta(\tilde{q'}, a_r) = \delta(\tilde{q}, v)$. Inductive assumption assures that $\delta(q_{r-k+1}, u)$ is correctly defined, hence it is possible to guarantee that $u \in V_{k-1}(\tilde{q}) \cap V_{k-1}(q_{r-k+1})$, being $\tilde{q}$ and q_{r-k+1} possibly equal. Nonetheless, the hypothesis imposes the equality: $D_{k-1}(\tilde{q}, u) = D_{k-1}(q_{r-k+1}, u)$. We know that v is in $D_{k-1}(\tilde{q}, u)$, hence v belongs also to $D_{k-1}(q_{r-k+1}, u)$, therefore $\delta(q_{r-k+1}, v)$ is defined and we finally get that $\delta(q_r, a_{r+1})$ is defined.

It remains to prove condition (14): $\delta(q_0, w) = q_m \in F$. $R_k(w) = a_{m-k+1}a_{m-k+2} \dots a_m \in \gamma_k$. Let us consider the states: $q_{m-k+1} = \delta(q_0, a_1 a_2 \dots a_{m-k+1})$ and $q_m = \delta(q_{m-k+1}, a_{m-k+2}a_{m-k+3} \dots a_m)$. The string $a_{m-k+1}a_{m-k+2} \dots a_m$ is in γ_k, therefore a string y of L exists such that $a_{m-k+1}a_{m-k+2} \dots a_m$ is its k-length suffix. Being y in L, a state $\tilde{q}$ exists: $\delta(\tilde{q}, a_{m-k+2}a_{m-k+3} \dots a_m) = q_f \in F$. In particular: $a_{m-k+2}a_{m-k+3} \dots a_m \in V_{k-1}(q_{m-k+1}) \cap V_{k-1}(\tilde{q})$. Basing on lemma 2, the state $\delta(q_{m-k+1}, a_{m-k+2}a_{m-k+3} \dots a_m)$ is equivalent to $\delta(\tilde{q}, a_{m-k+2}a_{m-k+3} \dots a_m)$. Therefore, q_m is equivalent to q_f: q_m belongs to F.
□

Theorems 2 and 3 lead us to directly obtain the main result of this section, characterizing $LT_k s.s.$

Theorem4 A language L, accepted by a DFA M, is $LT_k s.s.$ iff $D_{k-1}(q_i, x) = D_{k-1}(q_h, x)$ for any $q_i, q_h \in Q$ and for any $x \in V_{k-1}(q_i) \cap V_{k-1}(q_h)$.

The following corollary is a straightforward consequence of theorem 3 and corollary 1:

Corollary2 A language L, accepted by the reduced DFA M, is $LT_k s.s.$ iff $\delta(q_i, x) = \delta(q_h, x)$ for any $q_i, q_h \in Q$ and for any $x \in V_{k-1}(q_i) \cap V_{k-1}(q_h)$.

5. Order of a $LT_k s.s.$ language

If L is in $LT_k s.s.$, clearly it is also in $LT_{k'} s.s.$, with $k' > k$. However, in general it is not true that L is in $LT_{k''} s.s.$ where $k'' < k$. The order of a language is defined as the minimum value of the parameter k such that L is in $LT_{k_{min}} s.s.$ In the more general case of Locally Testable class (LT), it is known that the analogous problems is NP-Hard[5]. As concerns $LT s.s.$ languages, no explicit result is available in literature about the complexity class of the problem. Our approach consists in fixing the parameter k and in formulating the decision problem corresponding to the question: "*Is L in $LT_k s.s.$?*". In previous section, a decidibility procedure was proved. Hence it is possible to immediately obtain a decidibility algorithm for the different problem: "*Is L in $LT s.s.$?*". Such a result bases on theorem 1, that expresses an upper bound to use when ascertaining local testability.

The approach we adopted presents the advantage that the same procedure can be employed for two different problems, respectively a decidibility and an optimization one. It allows us to decide if L is in $LT s.s.$, basing on theorem 1, and secondly to evaluate k_{min}, such that L is in $LT_{k_{min}} s.s.$, i.e. the order of L.

Let us consider a language L and one of its accepting automata, comprising n states. If L is in $LT s.s.$, then it necessarily is in $LT_{n+2} s.s.$ (theorem 1). It is possible to test whether such a condition holds by applying the result we proved about $LT_k s.s.$ decidibility (theorem 4) considering $k = n+2$. If the response is negative, then it means L is not $LT_k s.s.$ for any value of k. In case the response is affirmative, this proves L is $LT s.s.$, thus it is possible to proceed to determine the order of L. By iterating the test expressed in theorem 4, and decreasing each time the value of parameter k by unit, the minimal value k_{min} is produced such that $L \in LT_{k_{min}}$, as illustrated in the fragment of pseudo code below.

```
int order(automaton M=(N,E))
{
int upper_bound      =|N|+2;
int k                =upper_bound;
boolean is_k_testable=TRUE;

for(;k>0;--k)
  {
     is_k_testable=test_condition(M,k);
     /* verify condition of theorem 4*/

     if((!is_k_testable)&&(k==upper_bound))
```

```
            return(-1);/*M in not in LTs.s.
                         for any value of k*/

        else if(!is_k_testable)
                 /*M is in LTs.s., hence the
                   order will be evaluated*/

            return(k+1);/*Not possible to
                         decrease k further*/
    };
return(1);/*M is 1-testable: M is the whole
            free monoid*/
}
```

6. Conclusions

After introducing the concept of local testability in a general context, the paper focused directly on the specific case of locally testable string languages in a strict sense. Interest in such a class of formal languages is motivated not only on a theoretical but also on an applied level, because of links to parallel parsing and error handling.

Through the formulation of a set of lemmata and theorems, a decidibility result was obtained for $LT_k s.s.$ property. On an operative level it provides a constructive algorithm to utilize on an accepting automaton M of language L. The analyzed procedure tries to frame the extensively studied concept of local testability in a different perspective. It aims at capturing $LT_k s.s.$ in a direct and general manner without employing the algebraic properties of the syntactic monoid. Moreover, it does not impose minimality constraints on M.

As a direct consequence of the developed algorithm, it was possible to easily relate $LT_k s.s.$ decidibility to $LT s.s.$ decidibility and to the problem of finding the order of $LT s.s.$ language.

The adopted approach does not rely on algebraic concepts. Hence, this aspect could be of help to investigate the extension of the exposed considerations to the case of tree languages.

7. Acknowledgments

We are grateful to Mr. P. E. Purcell for his encouragement to write this paper and for the precious editing suggestions he gave us.

参 考 文 献

1) J. A. Brzozowski, I. Simon. *Characterization of locally testable events.* Discrete Mathematics, Vol. 4, 1973, pp. 243-271
2) J. A. Brzozowski. *Hierarchies of aperiodic languages.* R.A.I.R.O. Information Théorique (vol. 10, No. 8, août 1976, pp. 33-49)
3) R. S. Cohen, J. A. Brzozowski. *Dot-depth of star-free events.* J. Computer and System Sc., Vol. 5, 1971, pp. 1-16
4) S. Crespi-Reghizzi, M. A. Melkanoff, L. Lichten. *The use of grammatical inference for designing programming languages.* Comm. ACM 16, 2 (Feb. 1973), pp. 83-90
5) S. M. Kim, R. McNaughton, R. McCloskey. *A polynomial time algorithm for local testability problem of deterministic finite automata.* I.E.E.E. Trans. Comput., 40, 1991, pp. 1087-1093
6) S. M. Kim, R. McNaughton. *Computing the order of a locally testable automaton.* SIAM J. Comput. Vol. 23, No. 6, pp. 1193-1215, December 1994
7) A.M. Magnaghi. *La testabilitá locale nelle sinstasi dei linguaggi artificiali* Master Thesis, Adviser Prof. S. Crespi-Reghizzi, Politecnico di Milano, 1996
8) R. McNaughton, S. Papert. *Counter-free automata.* M.I.T. Press, Cambridge, MA, 1971
9) M. Perles, O. Rabin, E. Shamir. *The theory of definite automata.* I.E.E.E. Trans. Electronic Computers EC-12, 1963, pp.233-243
10) A. Potthoff, W. Thomas. *Regular tree languages without unary symbols are strar free.* in 9th International Conference on Fundamentals of Computation Theory, Zoltán Ésik, ed., Lecture Notes in Compute Science 710 (1993), pp. 396-405
11) M. Steinby. *A theory of language varieties.* in Tree Automata and Languages, M. Nivat and A. Podelski (editors), Elsevier Publ., 1992, pp. 57-81
12) Y. Zalcstein. *Locally testable languages.* J. Computer and System Sc., Vol. 6, 1972, pp. 151-167