Proceedings of the International Workshop, Automata for Cellular and Molecular Computing, MTA SZTAKI, Budapest, pages 120 - 128, 2007.

On Local Testability in Watson-Crick Finite Automata*

José M. Sempere

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n, 46022 Valencia, Spain
jsempere@dsic.upv.es

Abstract

Watson-Crick finite automata were first proposed in [2] inspired by formal language theory, finite states machines and some ingredients from DNA computing such as working with molecules as double stranded complementary strings. Here, we define different kinds of local testability in this model. Mainly, we will explore local testability in the upper (lower) strand and in the double strand.

1 Introduction

Watson-Crick finite automata (WKFA) [2] is a good example of how DNA biological properties can be adapted to propose computation models in the framework of DNA computing. A recent survey on WKFA has been published in [1]. The WKFA model works with double strings inspired by double-stranded molecules with a complementary relation between symbols (here, inspired by classical complementary relation between nucleotides A-T and C-G). Different restriction over the model have been proposed, mainly devoted to restrict the number of final states (i.e., *all final* and *stateless* WKFA) and the way of processing the upper and lower string (i.e., *1-limited* and *simple* WKFA). Here we propose a different characterization of the model based on a classical concept of formal language theory such as local testability.

Local testable languages were first defined by McNaughton and Papert [5]. These languages have been widely studied in the framework of learning systems (i.e., [3, 10]), DNA and protein analysis (i.e., [12, 13]) and formal languages and semigroups (i.e., [6]), among others.

Here, we will introduce local testability in different ways. First, we will introduce a representation theorem for languages accepted by WKFA, which allows us to study WKFA through linear and even linear languages. Then, we will study two possibilities of defining local testability: in the upper (lower) strand and in the double strand. Finally, we will give some guidelines for future works.

*Work partially supported by the Generalitat Valenciana under research project GV06/068

2 Basic Concepts and Notation

In this section we will introduce basic concepts from formal language theory according to [4, 8] and from DNA computing according to [7].

Formal language theory

An alphabet Σ is a finite nonempty set of elements named symbols. A string defined over Σ is a finite ordered sequence of symbols from Σ. The infinite set of all the strings defined over Σ will be denoted by Σ^*. Given a string $x \in \Sigma^*$ we will denote its length by $|x|$. The empty string will be denoted by λ and Σ^+ will denote $\Sigma^* - \{\lambda\}$. Given a string x we will denote by x^r the reversal string of x. A language L defined over Σ is a set of strings from Σ. Finally, $\Sigma^{\leq k}$ will denote the set of strings with length less than or equals to k and Σ^k will denote the set of strings with length equals to k.

A grammar is a construct $G = (N, \Sigma, P, S)$ where N and Σ are the alphabets of auxiliary and terminal symbols with $N \cap \Sigma = \emptyset$, $S \in N$ is the *axiom* of the grammar and P is a finite set of productions in the form $\alpha \to \beta$. The language of the grammar is denoted by $L(G)$ and it is the set of terminal strings that can be obtained from S by applying symbol substitutions according to P. Formally, $w_1 \underset{G}{\Rightarrow} w_2$ if $w_1 = u\alpha v$, $w_2 = u\beta v$ and $\alpha \to \beta \in P$. We will denote by $\overset{*}{\underset{G}{\Rightarrow}}$ the reflexive and transitive closure of $\underset{G}{\Rightarrow}$.

We will say that a grammar $G = (N, \Sigma, P, S)$ is *right linear* (regular) if every production in P is in the form $A \to uB$ or $A \to w$ with $A, B \in N$ and $u, w \in \Sigma^*$. The class of languages generated by right linear grammars coincides with the class of regular languages and will be denoted by $\mathcal{REG}$. We will say that a grammar $G = (N, \Sigma, P, S)$ is *linear* if every production in P is in the form $A \to uBv$ or $A \to w$ with $A, B \in N$ and $u, v, w \in \Sigma^*$. The class of languages generated by linear grammars will be denoted by $\mathcal{LIN}$. We will say that a grammar $G = (N, \Sigma, P, S)$ is *even linear* if every production in P is in the form $A \to uBv$ or $A \to w$ with $A, B \in N$, $u, v, w \in \Sigma^*$ and $|u| = |v|$. The class of languages generated by even linear grammars will be denoted by $\mathcal{ELIN}$. A well known result from formal language theory is the inclusions $\mathcal{REG} \subset \mathcal{ELIN} \subset \mathcal{LIN}$.

A homomorphism h is defined as a mapping $h : \Sigma \to \Gamma^*$ where Σ and Γ are alphabets. We can extend the definition of homomorphisms over strings as $h(\lambda) = \lambda$ and $h(ax) = h(a)h(x)$ with $a \in \Sigma$ and $x \in \Sigma^*$. Finally, the homomorphism over a language $L \subseteq \Sigma^*$ is defined as $h(L) = \{h(x) : x \in L\}$.

Local testability

Here, we will introduce the definition of local testability and local testability in the strict sense. For any string $x \in \Sigma^*$ and any integer value $k > 0$, the testability vector $v_k(x)$ is defined by the tuple $(i_k(x), t_k(x), f_k(x))$ where

$$i_k(x) = \begin{cases} x, & \text{if } |x| < k \\ u : x = uv, |u| = k - 1 & \text{if } |x| \geq k \end{cases}$$

$$f_k(x) = \begin{cases} x, & \text{if } |x| < k \\ v : x = uv, |v| = k-1 & \text{if } |x| \geq k \end{cases}$$

$$t_k(x) = \{v : x = uvw, u, w \in \Sigma^* \wedge |v| = k\}.$$

We will define the equivalence relation $\equiv_k$ in $\Sigma^* \times \Sigma^*$ as $x \equiv_k y$ iff $v_k(x) = v_k(y)$. It has been proved in [5] that $\equiv_k$ is a finite index relation and that $\equiv_k$ covers $\equiv_{k+1}$.

So, we will say that any language L is k-testable iff it is defined as the union of some equivalence classes of $\equiv_k$. In addition, L is local testable iff it is k-testable for any integer value $k > 0$. The family of k-testable languages will be denoted by $k - \mathcal{LT}$ while $\mathcal{LT}$ will denote the class of testable languages.

A different kind of testability is the so called testability in the strict sense which was again proposed in [5]. Here, for any alphabet Σ we will take the sets $I_k, F_k \subseteq \Sigma^{\leq k-1}$ and $T_k \subseteq \Sigma^k$. Then, a language L is said to be k-testable in the strict sense if the following equation holds

$$L \cap \Sigma^{k-1}\Sigma^* = (I_k\Sigma^*) \cap (\Sigma^* F_k) - (\Sigma^* T_k \Sigma^*).$$

Observe that, according to the last equation, any word in L with length greater than or equals to $k-1$ begins with a segment in I_k, ends with a segment in F_k and has no segment from T_k. Any language L is locally testable in the strict sense iff it is k-testable in the strict sense for any $k > 0$. The family of k-testable languages in the strict sense will be denoted by $k - \mathcal{LTSS}$ while $\mathcal{LTSS}$ will denote the class of testable languages in the strict sense.

It has been proved that $k - \mathcal{LT}$ is the boolean closure of $k - \mathcal{LTSS}$ [14]. Finally, it can be easily proved that both classes $k - \mathcal{LT}$ and $k - \mathcal{LTSS}$ are subclasses of $\mathcal{REG}$.

Watson-Crick finite automata

Given an alphabet $\Sigma = \{a_1, \cdots, a_n\}$, we will use the symmetric (and injective) relation of complementarity $\rho \subseteq \Sigma \times \Sigma$. For any string $x \in \Sigma^*$, we will denote by $\rho(x)$ the string obtained by substituting the symbol a in x by the symbol b such that $(a, b) \in \rho$ (remember that ρ is injective) with $\rho(\lambda) = \lambda$.

Given an alphabet Σ, a *sticker* over Σ will be the pair (x, y) such that $x = x_1 v x_2$, $y = y_1 w y_2$ with $x, y \in \Sigma^*$ and $\rho(v) = w$. The sticker (x, y) will be denoted by $\binom{x}{y}$. A sticker $\binom{x}{y}$ will be a complete and complementary *molecule* if $|x| = |y|$ and $\rho(x) = y$. A complementary and complete molecule $\binom{x}{y}$ will be denoted as $\begin{bmatrix} x \\ y \end{bmatrix}$. Obviously, any sticker $\binom{x}{y}$ or molecule $\begin{bmatrix} x \\ y \end{bmatrix}$ can be represented by $x\#y^r$ where $\# \notin \Sigma$. Here, we will use $x\#y^r$ instead of $x\#y$ due to the grammar construction that we will propose in the following. Furthermore, inspired by DNA structure $x\#y^r$ represents the upper and lower nucleotide strings within the same direction $3' - 5'$ (or $5' - 3'$).

Formally, an *arbitrary* WK finite automaton is defined by the tuple $M = (V, \rho, Q, s_0, F, \delta)$, where Q and V are disjoint alphabets (states and symbols), ρ is a symmetric (and injective) relation of complementarity between symbols of V, s_0 is the initial state, $F \subseteq Q$ is a set of final states and $\delta : Q \times \binom{V^*}{V^*} \to \mathcal{P}(Q)$ (which denotes the power set of Q, that is the set of all possible subsets of Q).

The language of complete and complementary molecules accepted by M will be denoted by the set $L_m(M)$, while the upper strand language accepted by M will be denoted by $L_u(M)$ and defined as the set of strings x such that M, after analyzing the molecule $\begin{bmatrix} x \\ y \end{bmatrix}$ enters into a final state.

A Representation Theorem

Now, given any WKFA M, we will introduce a representation theorem for the languages $L_m(M)$ and $L_u(M)$. First, observe that any double string $\binom{x}{y}$ can be represented by the string $x\#y^r$. Then, the following result holds.

Theorem 1 *(Sempere, [11]) Let $M = (V, \rho, Q, s_0, F, \delta)$ be an arbitrary WK finite automaton. Then there exists a linear language L_1 and an even linear language L_2 such that $L_m(M) = L_1 \cap L_2$.*

The construction for L_1 and L_2 proposed in the theorem is defined as follows. First, the grammar $G_1 = (N, V, P, s_0)$ where $N = Q$, s_0 is the axiom of the grammar and P is defined as

- If $q \in F$ then $q \to \# \in P$.
- If $p \in \delta(q, \binom{x_1}{x_2})$ then $q \to x_1\ p\ x_2^r \in P$.

The language L_2 is defined by the grammar $G_2 = (\{S\}, V, P, S)$ where P is defined as follows

- $S \to \# \in P$.
- For every pair of symbols $a, b \in V$, such that $(a, b) \in \rho$, $S \to aSb \in P$.

It can be easily proved that $L(G_2) = \{x_1\#x_2^r \in V^* : |x_1| = |x_2| \text{ and } \rho(x_1) = x_2\}$. That is, L_2 can be established as the set of complete and complementary molecules $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ in the form $x_1\#x_2^r$.

From L_1 and L_2 it is clear that $L_1 \cap L_2$ is the set of complete and complementary molecules accepted by M in the form $x\#y^r$.

In order to characterize the upper strand language we will provide the following result.

Corollary 1 *(Sempere, [11]) Let $M = (V, \rho, Q, s_0, F, \delta)$ be an arbitrary WK finite automaton. Then $L_u(M)$ can be expressed as $g(h^{-1}(L_1 \cap L_2) \cap R)$ with L_1 being a linear language, L_2 an even linear language, R a regular language and g and h homomorphisms.*

3 Local Testability in Watson-Crick Finite Automata

In this section, we will introduce local testability in the upper or lower strand, and in the double strand of the WKFA model. Given that the languages accepted by arbitrary WKFA can be represented by linear and even linear languages, we will introduce two reductions from these language classes to the class $\mathcal{REG}$.

The first transformation, the so called σ operator, was first introduced in [9] and it was applied for the definition of local testable even linear languages in [10]. It is defined inductively as follows: $\sigma : \Sigma^* \to (\Sigma \times \Sigma)^*(\Sigma \cup \{\lambda\})$ with

1. $\sigma(\lambda) = \lambda$,
2. $(\forall a \in \Sigma)\ \sigma(a) = a$,
3. $(\forall a, b \in \Sigma)\ (\forall x \in \Sigma^*)\ \sigma(axb) = [ab]\sigma(x)$.

The operation σ is applied over languages as $\sigma(L) = \{\sigma(x) : x \in L\}$.

The inverse transformation σ^{-1} can be easily deduced from σ. It has been proved that for every even linear language L, $\sigma(L)$ is regular [9].

The second transformation is a grammatical construction that transforms every linear grammar into an even linear one. It is defined as follows.

Let $G_1 = (N, \Sigma, P, S)$ be a linear grammar. Then $G_2 = (N, \Sigma \cup \{*\}, P', S)$ is an even linear grammar where the productions of P' are defined as follows.

- If $A \to w \in P$ then $A \to w \in P'$.
- If $A \to uBv \in P$ with $|u| = |v|$, then $A \to uBv \in P'$.
- If $A \to uBv \in P$ with $|u| < |v|$, then $A \to u *^{|v|-|u|} Bv \in P'$.
- If $A \to uBv \in P$ with $|u| > |v|$, then $A \to uBv*^{|u|-|v|} \in P'$.

The last grammar is an even linear one and it can be easily proved that $g(L(G_2)) = L(G_1)$ where g is a morphism such that $g(*) = \lambda$ and $g(a) = a$ for every $a \in \Sigma$.

Local testability in the double strand

We will take the representation proposed in theorem 2.1. So, any molecule $\begin{bmatrix} x \\ y \end{bmatrix}$ can be represented by $x\#y^r$. Let us take G_1 as the linear grammar proposed in the theorem and let us take G_2 as the transformed even linear grammar corresponding to G_1. Obviously, for any string $x\#y^r$ of $L(G_1)$ we obtain a string $u\#v$ in $L(G_2)$ such that $g(u)\#g(v) = x\#y^r$, where g is the morphism defined before.

Now, we can work with G_2 and we apply the transformation σ over $L(G_2)$. Observe that $\sigma(L(G_2))$ is regular.

Example 1 *Let $M = (V, \rho, Q, s_0, F, \delta)$ be the WKFA defined by the following transitions*

$$\delta(q_0, \begin{pmatrix} a \\ \lambda \end{pmatrix}) = \{q_a\}, \quad \delta(q_a, \begin{pmatrix} a \\ \lambda \end{pmatrix}) = \{q_a\}, \quad \delta(q_a, \begin{pmatrix} b \\ a \end{pmatrix}) = \{q_b\},$$

$$\delta(q_b, \begin{pmatrix} b \\ a \end{pmatrix}) = \{q_b\}, \quad \delta(q_b, \begin{pmatrix} c \\ b \end{pmatrix}) = \{q_c\}, \quad \delta(q_c, \begin{pmatrix} c \\ b \end{pmatrix}) = \{q_c\},$$

$$\delta(q_c, \begin{pmatrix} \lambda \\ c \end{pmatrix}) = \{q_f\}, \quad \delta(q_f, \begin{pmatrix} \lambda \\ c \end{pmatrix}) = \{q_f\}.$$

Let us take q_f as the final state, q_0 as the initial stated and the complementarity relation $\rho = \{(a,a),(b,b),(c,c)\}$. Then, every complete and complementary molecule accepted by M takes the form $\begin{bmatrix} a^n b^n c^n \\ a^n b^n c^n \end{bmatrix}$ with $n \geq 1$.

Now, the representation linear grammar G_M, according to M is defined by the following productions (take q_0 as the axiom)

$$\begin{array}{ll} q_0 \to aq_a, & q_a \to aq_a \mid bq_ba, \\ q_b \to bq_ba \mid cq_cb, & q_c \to cq_cb \mid q_dc, \\ q_d \to q_dc \mid \#. & \end{array}$$

The corresponding even linear grammar is the following

$$\begin{array}{ll} q_0 \to aq_a*, & q_a \to aq_a* \mid bq_ba, \\ q_b \to bq_ba \mid cq_cb, & q_c \to cq_cb \mid *q_dc, \\ q_d \to *q_dc \mid \#. & \end{array}$$

Finally, we can provide the following right linear grammar to obtain the transformation σ over the last grammar

$$\begin{array}{ll} q_0 \to [a*]q_a, & q_a \to [a*]q_a \mid [ba]q_b, \\ q_b \to [ba]q_b \mid [cb]q_c, & q_c \to [cb]q_c \mid [*c]q_d, \\ q_d \to [*c]q_d \mid \#. & \end{array}$$

Observe that the last grammar generates the language defined as $L = \{[a*]^n[ba]^m[cb]^p[*c]^q\# : n,m,p,q \geq 1\}$. Then, if we take the morphism g with $g(*) = \lambda$ and $g(d) = d$ for every $d \in \{a,b,c,\#\}$ we can obtain $g(\sigma^{-1}(L)) = \{a^n b^m c^p \# c^q b^p a^m : n,m,p,q \geq 1\}$ which, together with the complementary relation ρ, corresponds to the language accepted by M.

So, the definition of local testability (in the strict sense) will be applied over the regular language obtained by the result $\sigma(L(G_M))$ for any WKFA M. Observe that every transformed language in $k-\mathcal{LT}$ ($k-\mathcal{LTSS}$) has a corresponding local testable language defined by the transitions of the WKFA.

Local testability in the upper and lower strand

Now, we will deal only with the upper (lower) strand. Observe that, the definition of the WKFA transitions can be transformed into FA transitions by taking the upper or lower strand (i.e., the transition $p \in \delta(q, \binom{x}{y})$ implies that $p_u \in \delta_u(q, x)$ and $p_l \in \delta_l(q, y)$). So, for every WKFA we can obtain two different finite automata which control the transitions in the upper and lower strands. Here, we will work with *simple* WKFA [7]. We will say that a WKFA is *simple* if for every transition $\delta(q, \binom{x}{y})$ $x = \lambda$ or $y = \lambda$. It has been proved that simple WKFA are normal forms for arbitrary WKFA. That is, for every arbitrary WKFA there exists an equivalent simple WKFA. Furthermore, we can work with the so called 1*limited*WKFA which are simple WKFA where every transition is performed by analyzing only one symbol every time.

Now, we will obtain finite automata from arbitrary 1*limited* WKFA through the following construction. Let $M = (V, \rho, Q, s, F, \delta)$ be an arbitrary 1*limited*WKFA. Then, we can define the finite automaton $A_u = (Q, V, \delta_u, s, F)$, where δ_u is defined as follows

1. $p \in \delta_u(q, a)$ if and only if $p \in \delta(q, \binom{a}{\lambda})$,
2. $p \in \delta_u(q, \lambda)$ if and only if $p \in \delta(q, \binom{\lambda}{a})$.

We can define the finite automaton $A_l = (Q, V, \delta_l, s, F)$ where δ_l is defined as follows

1. $p \in \delta_l(q, a)$ if and only if $p \in \delta(q, \binom{\lambda}{a})$,
2. $p \in \delta_l(q, \lambda)$ if and only if $p \in \delta(q, \binom{a}{\lambda})$.

Example 2 *Let us take the WKFA of example 3.1. Then A_u is defined through the following transitions*

$$\begin{array}{lll} \delta_u(q_0, a) = \{q_a\}, & \delta_u(q_a, a) = \{q_a\}, & \delta_u(q_a, b) = \{q_{bb}\}, \\ \delta_u(q_{bb}, \lambda) = \{q_b\}, & \delta_u(q_b, b) = \{q_{bbb}\}, & \delta_u(q_{bbb}, \lambda) = \{q_b\}, \\ \delta_u(q_b, c) = \{q_{cc}\}, & \delta_u(q_{cc}, \lambda) = \{q_c\}, & \delta_u(q_c, c) = \{q_{ccc}\}, \\ \delta_u(q_{ccc}, \lambda) = \{q_c\}, & \delta_u(q_c, \lambda) = \{q_f\}. & \end{array}$$

*In the previous definitions, the states q_{bb}, q_{bbb}, q_{cc} and q_{ccc} have been introduced in order to obtain an equivalent 1*limited*WKFA from the one proposed initially. In this case $L(A_u) = a^+b^+c^+$. The same holds for $L(A_l)$.*

Observe that, in both automata A_u and A_l, the empty transitions correspond to the case that the WKFA is working in the other strand, so the finite automata ignores all the movements in that way.

Now, the first definitions for local testability come from a natural way of looking up to the FA A_u and A_l. We will say that a 1*limited*WKFA is upper (lower) locally testable (in the strict sense) if the language accepted by A_u (resp. A_l) is locally testable (in the strict sense). Observe that this definition implies the existence of different classes of languages accepted by WKFA which have local testability. These classes are defined as follows

- the class $k - \mathcal{LT}_u$ of languages accepted by 1*limited*WKFA which have k-local testability in the upper strand,
- the class $k - \mathcal{LTSS}_u$ of languages accepted by 1*limited*WKFA which have k-local testability in the strict sense in the upper strand,
- the class $k - \mathcal{LT}_l$ of languages accepted by 1*limited*WKFA which have k-local testability in the lower strand,
- the class $k - \mathcal{LTSS}_u$ of languages accepted by 1*limited*WKFA which have k-local testability in the strict sense in the lower strand.

We can make a step further the definition of a new kind of local testability in every strand by introducing a combination of testability classes considered up to now in an isolated way. Let us take the finite automata A_l and A_u proposed before. Observe that every state in the previous automata defines an equivalence class according to $\equiv_k$ defined in section 2. Now, remember that the relation $\equiv_{k-1}$ covers $\equiv_k$. So, if $L(A_l)$ is in $j - \mathcal{LT}$, then $L(A_l)$ belongs to $k - \mathcal{LT}$ for every $j \leq k$. The same holds for A_u. So, we can combine different equivalence classes in the upper and the lower strand and they define new classes $(k, j) - \mathcal{LT}$ of languages accepted by 1*limited*WKFA which have k-local testability in the upper strand and j-local testability in the lower strand, and the class $(k, j) - \mathcal{LTSS}$ of languages accepted by 1*limited*WKFA which have k-local testability in the strict sense in the upper strand and j-local testability in the strict sense in the lower strand.

4 Conclusions and Future Work

We have presented different ways of introducing local testability in WKFA. The new definitions come from a previous representation result. The new classes inherit the properties of local languages defined in a classical way. Anyway, there exist different relations which should be explored between the language classes defined in the double strand and in every strand separately. In addition, the relation between the language classes defined for upper and lower strand simultaneously should be explored too.

Furthermore, the relation between languages accepted by locally testable WKFA and arbitrary languages should be explored in order to test the power of local testability in these models. These issues will be investigated in future works.

References

[1] E. Czeizler, E. Czeizler. A Short Survey on Watson-Crick Finite Automata. *Bulletin of the EATCS No. 88*, pages 104-119. February, 2006.

[2] R. Freund, G. Păun, G. Rozenberg, A. Salomaa. Watson-Crick finite automata In *Proceedings of DNA Based Computers III DIMACS Workshop (June, 1997)*, pages 297-327. The American Mathematical Society. 1999.

[3] T. Head, S. Kobayashi, T. Yokomori. Locality, Reversibility, and Beyond: Learning Languages from Positive Data. In *Proceedings of 9th International Conference ALT98 (October, 1998)*, pages 191-204. LNCS 1501, Springer, 1998

[4] J. Hopcroft, J. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison Wesley Publishing Co.,1979.

[5] R. McNaughton, S. Papert. *Counter-free automata.* MIT Press, 1971.

[6] J.E. Pin. *Varieties of Formal Languages.* Plenum Publishing Co., 1986.

[7] Gh. Păun, G. Rozenberg, A. Salomaa. *DNA Computing. New computing paradigms.* Springer, 1998

[8] G. Rozenberg, A. Salomaa, editors. *Handbook of Formal Languages, Vol. 1.* Springer, 1997.

[9] J. M. Sempere, P. García. A Characterization of Even Linear Languages and its Application to the Learning Problem. In *Proceedings of the Second International Colloquium on Grammatical Inference, ICGI-94* (September, 1994), pages 28-44. LNAI 862, Springer-Verlag, 1994.

[10] J. M. Sempere, P. García. Learning Locally Testable Even Linear Languages from Positive Data. In *Proceedings of the 6th International Colloquium on Grammatical Inference ICGI 2002 (September, 2002)*, pages 225-236. LNAI 2484, Springer-Verlag, 2002.

[11] J. M. Sempere. A Representation Theorem for Languages accepted by Watson-Crick Finite Automata. *Bulletin of the EATCS No. 83*, pages 187-191. 2004.

[12] T. Yokomori, N. Ishida, S. Kobayashi. Learning local languages and its application to protein α-chain identification. In *Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences (January, 1994), Vol.5: Biotechnology Computing,* pages 113-122. IEEE, 1994.

[13] T. Yokomori, S. Kobayashi. Learning local languages and their application to DNA sequence analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1067-1079, 1998.

[14] Y. Zalcstein. Locally Testable Languages. *Journal of Computer and System Sciences*, 6:151-167, 1972.