

Learning in varieties of the form V^*LI from positive data[☆]

Pedro García, José Ruiz*

Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain

Received 30 March 2004; received in revised form 22 December 2005; accepted 9 May 2006

Communicated by D. Perrin

Abstract

Grammatical inference and finite semigroup theory are continuously developing. Nevertheless it seems that they are not interacting too much. We propose in this paper an inference method for languages that belong to varieties of the form V^*LI . Many well known families of languages like locally testable, reversible, dot-depth one, etc. are of that form. The method unifies existing algorithms for inference of some of those families and can be applied to some others that had not been inferred yet. It uses a result about the cascade product in which one of the factors is a transducer and the second is the automaton obtained by inferring the base case of the family using the transduction of the sample as input.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Grammatical inference; Varieties of languages

1. Introduction

Grammatical inference is the discipline that deals with learning of formal languages either from positive data (i.e. a sequence of the words of the language) or from a complete presentation (i.e. a sequence of the words of the language and of its complement).

Although grammatical inference using only positive samples started with the negative result [16] that establishes that even the family of regular languages is not identifiable from positive data in the limit, several non-trivial families of languages have been identified this way, and a characterization of those families has been proposed in [2]. Afterwards, algorithms that identify certain interesting subclasses of regular languages in the limit have been proposed in [3,15,31].

The aim of this paper is to propose a framework for inference of new families of regular languages from positive data. A regular or recognizable language can be described, among others, by a regular grammar, a finite automaton or a finite semigroup. If one sees the evolution of grammatical inference, one can observe that this discipline has grown closed to disciplines like automata or probability theories, but quite apart of the development of others, like the theory of finite semigroups.

The use and development by Eilenberg [8] of the concepts of *variety of languages* and *variety of finite semigroups* and the theorem that states the one to one correspondence between those concepts, opened new ways of research in the

[☆] Work partially supported by Ministerio de Ciencia y Tecnología under project TIC2003-09319-C03-02.

* Corresponding author.

E-mail addresses: pgarcia@dsic.upv.es (P. García), jruiz@dsic.upv.es (J. Ruiz).

area of classification of families of recognizable languages. The families which are the object of study in this paper are better described using the framework of Eilenberg's variety theorem.

The method we propose deals with families of languages that have in common the membership of their syntactic semigroup to varieties of the form $\mathbf{V}^*\mathbf{LI}$, where \mathbf{LI} is the variety of locally trivial semigroups.

Many well known varieties of languages correspond to varieties of semigroups of the form $\mathbf{V}^*\mathbf{LI}$. For example, the family of *Locally Testable Languages* (McNaughton [25]), which corresponds to the variety of locally idempotent and commutative semigroups (Brzozowski and Simon [7] and Zalcstein [37]) is $\mathbf{LJ}_1 = \mathbf{J}_1 * \mathbf{LI}$. We also have that the first level in the dot-depth hierarchy, \mathbf{B}_1 is $\mathbf{B}_1 = \mathbf{J} * \mathbf{LI}$, where \mathbf{J} is the family of \mathcal{J} -trivial semigroups, $\mathbf{B}_2 = \mathbf{PJ} * \mathbf{LI}$, where \mathbf{PJ} is the variety of monoids generated by $\mathcal{P}(M)$ for $M \in \mathbf{J}$, and in general $\mathbf{B}_n = \mathbf{V}_n * \mathbf{LI}$ (Straubing), where \mathbf{B}_n are the varieties of monoids of Brzozowski's hierarchy and \mathbf{V}_n are the varieties of Straubing's hierarchy ($\mathbf{V}_0 = \mathbf{I}$, $\mathbf{V}_1 = \mathbf{J}$).

Another variety for which this method can be used is the variety $\mathbf{G}_{\text{com}} * \mathbf{LI}$, which is the variety of semigroups whose local monoids are commutative groups. This variety corresponds with the variety of languages defined by the congruence in which two words are equivalent if there exist values of k and m such that both words have the same prefix and suffix of length $k - 1$ and the same set of segments of length k in the same number (counted modulo m).

As, in the process of inference, the words of the sample have to be read, some varieties in which the order of appearance of the segments or subwords is considered have proved to be useful in this area. Then, we will work with varieties like $\mathbf{R}_1 * \mathbf{LI}$, which is the variety of right locally testable languages [12] or $\mathbf{LR} = \mathbf{R} * \mathbf{LI}$. The languages belonging to this later variety of languages are generated by the congruence in which two words are equivalent if they have the same set of subwords of length k of segments of length m for given values of k and m , considering the order of appearance of those subwords.

Another families that will be shown to be inferrable using this method are the family of k -reversible languages [2] and k -testable languages in the strict sense [15].

2. Definitions and notation

In this section we will describe some facts about formal languages in order to make the notation understandable to the reader. For further details about the definitions, the reader is referred to [8,20,26].

2.1. Languages and automata

Let A be a finite alphabet and A^* the free monoid generated by A with concatenation as the binary operation and ε as neutral element. Any subset $L \subseteq A^*$ is called *language*, we will refer to its elements as *words* and the length of a word x will be denoted as $|x|$. Let A^k (resp. $A^{\leq k}$) be the set of words of length k (less than or equal to k) over A . The set of symbols that a word x contains is denoted as $\alpha(x)$.

Given $x \in A^*$, if $x = uvw$ with $u, v, w \in A^*$, then u (resp. w) is called *prefix* (resp. *suffix*) of x , whereas v is called a *segment* of x . The prefix (resp. suffix) of length k of x is denoted $i_k(x)$ (resp. $f_k(x)$). The set of segments of length k of x is denoted $t_k(x)$. The set of prefixes (resp. suffixes) of a word x will be denoted as $\text{Pr}(x)$ (resp. $\text{Suf}(x)$).

Let $L \subseteq A^*$ and \equiv be an equivalence relation defined in A^* . We say that \equiv saturates L , if L is the union of equivalence classes modulo \equiv . An equivalence relation is called a congruence if it is both-sides compatible with the operation of the monoid.

A *semiautomaton* is a triple (Q, A, δ) , where Q is a finite set of states, A is a finite alphabet and δ is the transition function,¹ a partial function that maps $Q \times A$ in Q , which can be extended to words by setting $\delta(q, \varepsilon) = q$ and $\delta(q, xa) = \delta(\delta(q, x), a)$, for every $q \in Q$, for every $x \in A^*$ and every $a \in A$. A *deterministic finite automaton (DFA)* is a quintuple $\mathcal{A} = (Q, A, \delta, q_0, F)$ where Q, A and δ are the same than before, whereas $q_0 \in Q$ is the initial state and $F \subseteq Q$ is the set of final states. The *reverse* of δ , denoted δ^r , is defined by $\delta^r(q, a) = \{q' : \delta(q', a) = q\}$. A word x is accepted by an automaton \mathcal{A} if $\delta(q_0, x) \in F$. The set of words accepted by \mathcal{A} is denoted by $L(\mathcal{A})$. A word u is a *k-follower* (resp. *k-leader*) of the state q in \mathcal{A} if $|u| = k$ and $\delta(q, u) \neq \emptyset$ (resp. $\delta^r(q, u^r) \neq \emptyset$).

Given a finite set of words S , the *prefix tree acceptor* of S is defined as the automaton $\mathcal{A} = (Q, A, \delta, q_0, F)$ where $Q = \text{Pr}(S)$, $q_0 = \varepsilon$, $F = S$ and $\delta(u, a) = ua$, for every $u, ua \in Q$.

¹ We use the notation $\delta(q, a)$ as it is used in the grammatical inference community. Semigroup theoretists use qa instead.

A *sequential machine* is a 5-tuple $M = (Q, A, B, \delta, \lambda)$ where Q, A and δ are defined in the same way as in a DFA, B is the output alphabet and the output function λ is a function that maps $Q \times A$ into B , which can be extended to $Q \times A^*$ by establishing $\lambda(q, \varepsilon) = \varepsilon$, and $\lambda(q, xa) = \lambda(q, x)\lambda(\delta(q, x), a)$. The function $t : A^* \rightarrow B^*$ realized by M is $t(x) = \lambda(q_0, x)$.

2.2. Semigroups and varieties

A *semigroup* is a couple (S, \cdot) , where S is a set and \cdot is an internal associative law defined on S . S is a *monoid* if \cdot has a neutral element $1 \in S$. An element $e \in S$ is called *idempotent* if $e^2 = e$. The set of idempotents of a semigroup S is denoted $E(S)$. For every idempotent e , eSe is a monoid called the *local monoid* associated to e . A semigroup S is a *subsemigroup* of T if there is an injective morphism $\eta : S \rightarrow T$. S is a *quotient* of T if there is a surjective morphism $\mu : T \rightarrow S$. A semigroup S *divides* a semigroup T ($S \prec T$) if S is a quotient of a subsemigroup of T .

A monoid M *recognizes* a language $L \subseteq A^*$ if there exists a monoid morphism $\mu : A^* \rightarrow M$ and a subset P of M such that $L = \mu^{-1}(P)$.

The *syntactic congruence* of a language L is the congruence \sim_L over A^* defined as $x \sim_L y$ if and only if for every $u, v \in A^*$, $(uxv \in L \Leftrightarrow uyv \in L)$. The *syntactic monoid* of L is the quotient monoid $M(L) = A^*/\sim_L$.

A *variety of finite semigroups (ordered)* is a family of semigroups (ordered semigroups) closed under division and under finite direct products. A *variety of languages* is a family of recognizable languages closed under boolean operations, inverse morphisms and quotients. If \mathbf{V} is a variety of monoids and A is an alphabet, we denote by $\mathcal{L}_V(A^*)$ the set of languages of A^* recognized by a monoid of \mathbf{V} . The *variety theorem* [8] states that the function that assigns to every variety \mathbf{V} the variety $\mathcal{L}_V(A^*)$ is a bijection.

We denote by \mathbf{LV} the class of semigroups which are locally in \mathbf{V} , that is, the semigroups S such that $eSe \in \mathbf{V}$, for every $e \in E(S)$.

A transformation semigroup is a pair $\mathcal{X} = (Q, S)$, where Q is a non-empty finite set and S is a set of functions from Q into Q which is a semigroup under the composition of functions. Given $q \in Q$ and $s \in S$, $s(q)$ is simply denoted as qs .

Given two transformation semigroups $\mathcal{X} = (Q, M)$ and $\mathcal{Y} = (P, N)$, the set of functions from $P \times N$ into M is denoted $M^{P \times N}$. The *wreath product* $\mathcal{X} \circ \mathcal{Y}$ is the transformation semigroup $(Q \times P, M^{P \times N} \times N)$ defined as follows: for every $(q, p) \in Q \times P$ and every $(f, n) \in M^{P \times N} \times N$, $(q, p)(f, n) = (qf(p, n), pn)$.

If $\mathcal{A} = (Q, A, \delta)$ is a semiautomaton, the pair $\mathcal{X} = (Q, S)$, where S is the set of all the functions $x^A : Q \rightarrow Q$, $x \in A^*$, defined as $x^A(q) = \delta(q, x)$, is a transformation semigroup.

Given an automaton $\mathcal{B} = (Q, B, \delta_1, q_0, F)$ and a sequential machine $\tau = (P, A, B, \delta_2, p_0, \lambda)$, the cascade product $\mathcal{B} * \tau$ is the automaton $\mathcal{B} * \tau = (Q \times P, A, \delta, (q_0, p_0), F \times P)$, with the transition function defined as $\delta((q, p), a) = (\delta_1(q, \lambda(p, a)), \delta_2(p, a))$. The transformation semigroup of the semiautomaton $(Q \times P, A, \delta)$ is a subsemigroup of $M \circ M(\tau)$, where M and $M(\tau)$ are the transformation semigroups of \mathcal{B} and τ , respectively.

Given two varieties of finite semigroups \mathbf{V} and \mathbf{W} , the product $\mathbf{V} * \mathbf{W}$ is the variety generated by all the wreath products $S \circ T$ for every $S \in \mathbf{V}$ and for every $T \in \mathbf{W}$.

An *ordered semigroup* (S, \leq) is a semigroup equipped with a stable order relation \leq on S . A *variety of finite ordered semigroups* is a class of finite ordered semigroups closed under the taking of ordered subsemigroups, ordered quotients and finite products [28].

We will next describe two important varieties of semigroups and languages that will be used throughout this paper, the *locally trivial* semigroups and the *\mathcal{J} -trivial* semigroups. Languages corresponding to this later variety are called the *piecewise testable languages*. We will also describe some other closely related families.

Given $x, y \in A^*$, we say that $x = a_1 a_2 \dots a_n$, with $a_i \in A, i = 1, 2, \dots, n$ is a *subword* of y , if $y = z_0 a_1 z_1 a_2 \dots a_n z_n$ with $z_i \in A^*, i = 0, 1, 2, \dots, n$.

We say that a language is *piecewise testable* if the two following equivalent conditions hold:

- It is saturated by a congruence in which, given an integer k , two words are equivalent if they have the same set of subwords of length smaller than or equal to k .
 - It is a boolean combination of languages of the form $A^* a_1 A^* a_2 \dots A^* a_n A^*$, where a_1, \dots, a_n are elements of A .
- It is well known [34] that the family of piecewise testable languages is the variety of languages $\mathcal{L}_{\mathbf{J}}(A^*)$, being \mathbf{J} the variety of \mathcal{J} -trivial monoids.

If we also consider the order of appearance of the subwords when we scan the prefixes of the words, the languages defined by this new congruence is $\mathcal{L}_R(A^*)$, being \mathbf{R} the variety of R -trivial monoids [6].

The variety of idempotent and commutative (resp. idempotent and $xyx = xy$) will be denoted as \mathbf{J}_1 (resp. \mathbf{R}_1).

A semigroup S is *locally trivial* if for every $s \in S$, $e \in E(S)$, we have $ese = e$. The corresponding variety of languages are the languages of the form $XA^*Y \cup Z$, where X , Y and Z are finite languages of A^* . They are also the boolean algebra generated by languages of the form uA^* and A^*u with $u \in A^+$.

2.3. Inference and learning in the limit

Grammatical inference is the discipline that deals with the efficient learning of formal languages either from examples or from examples and counterexamples. In this context, a *presentation* of a language L is a sequence of the words of L .

We use the model of learning called *identification in the limit* [16]. An algorithm A identifies a class of languages \mathcal{L} by means of hypothesis in \mathcal{H} in the limit if and only if for any $L \in \mathcal{L}$, on input of any presentation of L , the infinite sequence of hypothesis output by A converges to h such that $L(h) = L$, that is, there exists t_0 such that ($t \geq t_0 \Rightarrow h_t = h_{t_0} \wedge L(h_{t_0}) = L$), where h_t denotes the hypothesis emitted by A after processing t examples.

2.4. Some varieties of the form $\mathbf{V}^* \mathbf{LI}$

2.4.1. Locally testable languages and related families

A language is *locally testable* [25] if it is saturated by a relation in which given an integer k , two words of length greater than or equal to k are equivalent if they have the same prefix and suffix of length $k - 1$ and the same set of segments of length k . If we also consider the order of appearance of new segments when we scan the prefixes (resp. suffixes) of the words, the languages generated by this finer congruence are called *right* (resp. *left*) *locally testable* [12]. The variety of locally testable languages corresponds to the variety of locally idempotent and commutative semigroups [7,37].

2.4.2. Reversible languages

In [3] Angluin defines the family of k -reversible languages and proposes an algorithm that for each value of k and each positive sample S , outputs the smallest k -reversible language that contains S . This family has been used [22] to upper-best approximately identify the regular languages in the limit from positive data.

A *zero-reversible* automaton is a deterministic finite-state automaton with at most one final state such that $\delta(q, a) = \delta(q', a)$ implies that $q = q'$. A language is *k-reversible* if it is accepted by a deterministic automaton with the following property: for any pair of distinct states q_1 and q_2 , if $q_1, q_2 \in F$ or $q_1, q_2 \in \delta^r(q, a)$ there is no common k -leader of both q_1 and q_2 . A language is *zero-reversible* (resp. *k-reversible*) if it is accepted by a zero-reversible (resp. *k-reversible*) automaton. The family of zero-reversible (resp. *k-reversible*) languages (in the sense of Angluin) will be denoted as R_0 (resp. R_k).

The family of reversible languages is closed under intersection, but not closed under union and complement.

Pin [27,28] proposes a family which is an extension of the family of zero-reversible languages that is also closed under union. The cost is that the minimal automaton that recognizes a zero-reversible language is no longer zero-reversible. An extension of that family for any value of k has been recently proposed [29].

Definitions concerning these new families will be seen in Section 4.2. The family of k -reversible languages in the sense of Pin will be called Rev_k . We will show that $\mathbf{Rev}_k = \mathbf{Rev}_0 * \mathbf{LI}_{k+1}$.

2.4.3. Dot-depth one languages

Following [33], if k and m are integers and $v = (w_1, \dots, w_m)$ is an m -tuple of words of length k over A , we say that v appears in a word u if u can be written as $u = u_i w_i v_i$, $i = 1, \dots, m$, where u_i and v_i are words over A such that the sequence $|u_i|$ is strictly increasing.

We say that $u \sim_{m,k} v$ if:

- (1) u and v have the same prefix of length $k - 1$.
- (2) u and v have the same suffix of length $k - 1$.
- (3) The same m -tuples of words of length k appear in u and in v .

We say that a language is *dot-depth one* if one of the two following conditions hold:

- It is saturated by the congruence $\sim_{m,k}$ for some values of m and k .
 - It is a boolean combination of languages of the form $A^*w_1A^*w_2\dots A^*w_nA^*$, where w_1, \dots, w_n are words over A .
- Dot-depth one languages form a variety of languages. The corresponding variety of semigroups is known in the literature as \mathbf{B}_1 . We have that $\mathbf{B}_1 = \mathbf{J} * \mathbf{LI}$.

As $\mathbf{LR} = \mathbf{R} * \mathbf{LI}$, there is an extension of this later variety which corresponds with the variety of languages generated by the congruence in which two words x and y are equivalent if

- (1) $f_{k-1}(x) = f_{k-1}(y)$.
- (2) For every $u \in \text{Pr}(x)$ there exists $v \in \text{Pr}(y)$ such that $u \sim_{m,k} v$.
- (3) For every $u \in \text{Pr}(y)$ there exists $v \in \text{Pr}(x)$ such that $u \sim_{m,k} v$.

That is, x and y must have the same suffix of length $k - 1$ and the same set of subwords of length k of segments of length m for given values of k and m , and they must appear in the same order when we scan the prefixes of x and y .

2.4.4. Periodic locally testable languages

In [21] Knast defines the families of periodic locally testable languages. Given integers k and p , we say that a language L is *k-locally testable modulo p* if it is saturated by a congruence in which two words of length greater than $k - 1$ are equivalent if:

- (1) begin and end by the same segments of length $k - 1$;
- (2) they have the same segments of length k counted modulo p .

Note that in this definition, every segment of length k is considered. A segment not present in a word is considered as being a number of times multiple of p in that word.

A language is *periodic locally testable* if it is k -locally testable modulo p for some values of k and p .

Periodic locally testable languages correspond with the variety of locally commutative groups (\mathbf{LG}_{com}) [21]. We have that $\mathbf{LG}_{\text{com}} = \mathbf{G}_{\text{com}} * \mathbf{LI}$.

This family has also been defined and characterized in [36] through logic sentences having modular quantifiers and successor and equality as numerical predicates.

A family of languages very similar to this one, which may be interesting in pattern recognition has been defined in [11]. The difference with respect to the former one is that in this one, if a segment is not present in a word, it is not considered.

3. The general method

We recall the following definitions and theorems that will be used in the inference process:

Theorem 1 (Ginzburg and Rose, see Berstel [4]). *Let $\tau : A^* \rightarrow B^*$, be the sequential function realized by the transducer $\tau = (P, A, B, \delta_1, \lambda, p_0)$, let $\mathcal{A} = (Q, B, \delta_2, q_0, F')$, and let $L = L(\mathcal{A})$. The language $\tau^{-1}(L) \subseteq A^*$ is recognized by the cascade product $\mathcal{A} * \tau = (Q \times P, A, \delta, [q_0, p_0], F' \times Q)$, with the transition function defined as $\delta([q, p], a) = (\delta_2(q, \lambda(p, a)), \delta_1(p, a))$.*

Let M and $M(\tau)$ the transformation semigroups associated to \mathcal{A} and τ , respectively, then $\tau^{-1}(L)$ is recognized by $M \circ M(\tau)$. If $M \in \mathbf{V}$ and $M(\tau) \in \mathbf{W}$, then $\tau^{-1}(L) \in \mathcal{L}_{V * W}(A^*)$.

Definition 2. For $k > 0$, let $\tau_k = (Q, A, B, \delta, \lambda, q_0)$ be a sequential machine defined as $Q = \bigcup_{i=0}^{k-1} A^i$, $p_0 = \varepsilon$, $B = \bigcup_{i=1}^{k-1} \#^{k-i} A^i \cup A^k$ and for every $p \in Q$ and $a \in A$, the transition and output functions are, respectively, defined as:

$$\delta(p, a) = \begin{cases} pa & \text{if } |p| < k - 1 \\ f_{k-1}(pa) & \text{if } |p| = k - 1 \end{cases} \quad \text{and}$$

$$\lambda(p, a) = \begin{cases} \#^{k-|pa|} pa & \text{if } |p| < k - 1 \\ pa & \text{if } |p| = k - 1. \end{cases}$$

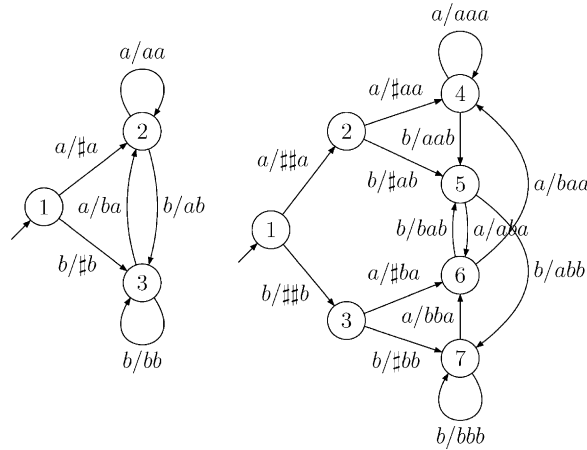


Fig. 1. Transducers used for inference for values $k = 2$ and 3 .

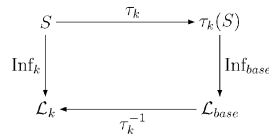


Fig. 2. Scheme of the inference method.

The sequential machine τ_k , for a given $k > 0$ and a word $x \in A^*$, outputs a word $\tau_k(x)$ whose symbols are the segments of length k of x , in order. Examples of τ_k for the values $k = 2$ and 3 and $A = \{a, b\}$ can be seen in Fig. 1.

The following definitions and propositions are due to Straubing, see [35]:

Definition 3. Let A be a finite alphabet, let $C = A^k$ and let \sim be an equivalence relation defined in C^* . We define an equivalence relation $w_1 \sim_k w_2$ in A^+ as follows:

- If $|w_1| < k$, $w_1 \sim_k w_2$ if and only if $w_1 = w_2$.
- If $|w_1|, |w_2| \geq k$, $w_1 \sim_k w_2$ if and only if:
 - $i_{k-1}(w_1) = i_{k-1}(w_2)$.
 - $f_{k-1}(w_1) = f_{k-1}(w_2)$.
 - $\tau_k(w_1) \sim \tau_k(w_2)$.

Proposition 4. If \sim is an congruence of finite index in $C^* = (A^k)^*$, then \sim_k is an congruence of finite index in A^+ .

Proposition 5. A^+ / \sim_k divides $C^* / \sim \circ A^+ / \mu_k$, where μ_k is the congruence that defines the variety \mathbf{LI}_k .

This proposition establishes that if \mathbf{V} is the variety of monoids corresponding to the variety of languages saturated by the congruence \sim in C^* , L is saturated by \sim_k if and only if L belongs to the variety of languages recognized by semigroups in $\mathbf{V} * \mathbf{LI}_k$.

The previous definitions and theorems permit us to propose an algorithm that, for every sample S , and a value of $k > 0$ obtains the smallest language of the desired variety of the form $\mathbf{V} * \mathbf{LI}_k$ that contains S .

The scheme of the method is depicted in Fig. 2. For each $x \in S$ we obtain $\tau_k(x)$, that is, the word whose symbols are the segments of length k of x , using a fixed transducer τ_k (for each k). Afterwards we infer the automaton \mathcal{A} , which recognizes the smallest language of the desired type (base case) that contains the transduced word and finally we realize the cascade product $\mathcal{A} \circ \tau_k$.

The only difficulty that we may have if we apply Theorem 1 to the inference problem, as we have done the proof for just one equivalence class, is the way of establishing the set of final states in the output automaton when we have

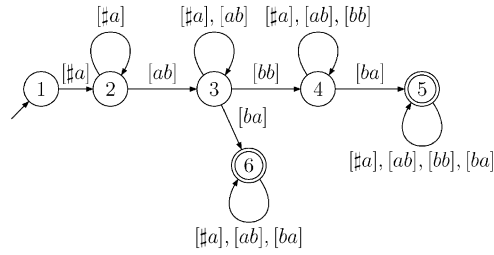


Fig. 3. Base case obtained for the sample $S = \{abba, aba\}$ and the value $k = 2$.

words belonging to several equivalence classes. The easiest solution is to set up $\delta(q_0, S) = \bigcup_{x \in S} \delta(q_0, x)$ as the set of final states of the automaton.

If, in the inference process, we obtain the smallest language of a certain type that contains the transduction of the sample for the base case, as both τ_k and τ_k^{-1} are bijective, it follows that the proposed scheme outputs the smallest language that contains the sample in the general case.

4. Examples

4.1. Smallest k -testable language that contains a positive sample S

Given an alphabet B and $u \in B^*$, we denote by $l(u)$ the list of the first occurrences of the symbols of B in the word u read from left to right. For example, if $u = abbabcba$, $l(u) = \langle a, b, c \rangle$. More formally, $l(u)$ can be recursively defined as follows:

- $l(\varepsilon) = \emptyset$ (empty list) and
- $l(ub) = \begin{cases} l(u) & \text{if } b \in l(u), \\ l(u) \wedge b & \text{otherwise} \end{cases}$ (\wedge means concatenation).

Definition 6. Given a sample S , let \mathcal{A} be the automaton $\mathcal{A} = (Q, B, \delta', q_0, F')$, where the alphabet is $B = \alpha(\tau_k(S)) = t_k(S)$, $Q = \{l(u) : u \in \text{Pr}(S)\}$, that is, every state is defined by an ordered list that contains the alphabet of the prefix of a word of the sample, $q_0 = l(\varepsilon)$, $F' = \{l(y) : y \in \tau_k(S)\}$ and the transition function defined as $\delta'(l(u), b) = l(ub)$ for every $l(u) \in Q$ and every $b \in B$.

Proposition 7. The automaton \mathcal{A} recognizes the smallest 1-right testable language that contains the sample $\tau_k(S)$.

Proof. Segments and subwords of length 1 of a word coincide, so applying the algorithm proposed in [13] for inference of piecewise testable languages for the value $k = 1$ to $\tau_k(S)$, the automaton \mathcal{A} is obtained. \square

4.1.1. Example

Let $S = \{abba, aba\}$ and let $k = 2$. The transduction of the sample gives us the set of words $\{\langle [\#a], [ab], [bb], [ba] \rangle, \langle [ab], [ba] \rangle\}$ over the alphabet $\{[\#a], [ab], [bb], [ba]\}$. Using this alphabet and applying the algorithm for the base case we obtain the automaton depicted in Fig. 3.

We recall that the transition function of the cascade product is $\delta([q, p], a) = (\delta_2(q, \lambda(p, a)), \delta_1(p, a))$, where δ_1 and λ are, respectively, the transition and the output function of the transducer, whereas δ_2 is the transition function of the base-case automaton. For example $\delta([2, 3], a) = (\delta_2(2, \lambda(3, a)), \delta_1(3, a)) = (\delta_2(2, ba), 2) = (5, 2)$.

After doing the cascade product by the automaton of Fig. 1 for $k = 2$, we obtain the automaton of Fig. 4, which recognizes the smallest 2 right-testable language that contains the word $abba$.

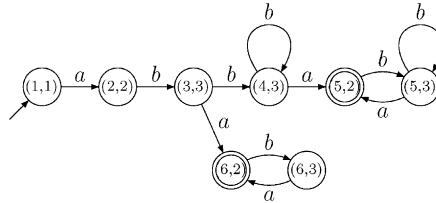


Fig. 4. Automaton that recognizes the smallest 2-right testable language that contains the sample $S = \{abba, aba\}$.

4.2. Smallest k -reversible language that contains a positive sample S

Pin [27] proposes a family which is an extension of the family of zero-reversible languages that is also closed under union. The cost is that the minimal automaton that recognizes a zero-reversible language is no longer zero-reversible. An extension of that family for any value of k has been recently proposed [29].

We recall the definitions concerning those families.

Definition 8. A reversible automaton [27] (in the sequel these type of automata will be called as zero-reversible) is a finite automaton in which each letter induces a partial one to one map from the set of states into itself.

Definition 9. A language L belongs to Rev_0 if it is accepted by a zero-reversible automaton.

Proposition 10. The family Rev_0 is closed under union and intersection, inverse morphisms and quotients (not under complement), so it is a positive variety of languages [28]. The corresponding variety of ordered monoids is $\mathbf{Rev}_0 = \mathbf{J}_1^- * \mathbf{G} = \mathbf{Ecom}^- = [x^w y^w = y^w x^w, 1 \leq x^w]$.

Note that Angluin’s definition of a zero-reversible automaton is more restricted in the sense that it also requires for the automaton to be deterministic and to have only one initial and one final states. It is easily seen that a zero-reversible language in the sense of Angluin is zero-reversible in the sense of Pin. By the other hand, as every finite language belongs to Rev_0 , this family is not identifiable from positive data in the limit.

4.2.1. k -Reversible automata and languages

Definition 11. A k -reversible automaton is a deterministic finite automaton such that for any states q, q_1 and q_2 and any symbol a with $\delta(q_1, a) = \delta(q_2, a) = q$, if there exist states q_3 and q_4 and a word $u \in A^k$ with $\delta(q_3, u) = q_1$ and $\delta(q_4, u) = q_2$ then $q_1 = q_2$.

Definition 12. A language L belongs to Rev_k if it is accepted by a k -reversible automaton.

Proposition 13. Let $L \in Rev_k$, then $\tau_{k+1}(L) \in Rev_0$.

Proof. Let $L = L(\mathcal{A})$, where \mathcal{A} is the automaton $\mathcal{A} = (P, A, \delta, p_0, F)$ and let $\tau_{k+1} = (Q, A, B, \delta', \lambda', \varepsilon)$. Let $\mathcal{B} = (P \times Q, B, \delta'', (p_0, \varepsilon), F \times (Q - \{\varepsilon\}))$, where the transition function δ'' is defined as $\delta''((p, q), b) = (\delta(p, a), \delta'(q, a))$, being $a \in A$ such that $\lambda'(q, a) = b$. It is easily seen that $L(\mathcal{B}) = \tau_{k+1}(L)$.

Let us see that $L(\mathcal{B}) \in Rev_0$ by contradiction. If $\tau_{k+1}(L)$ is not zero-reversible we would have in automaton \mathcal{B} , $\delta''((p_1, x_1), [x_1 b]) = \delta''((p_2, x_2), [x_2 b]) = (p, y)$ with $|x_1| = |x_2| = |y| = k$ and $[x_1 b] = [x_2 b]$, so $x_1 = x_2 = x$. This situation is pictured in Fig. 5.

Then, as seen in Fig. 6, in \mathcal{B} we will have the configuration $\delta''((p_0, \varepsilon), y[ax]) = (p_1, x)$, also $\delta''((p_0, \varepsilon), y'[a'x]) = (p_2, x)$ and $\delta''((p_1, x), [xb]) = \delta''((p_2, x), [xb]) = (p, f_k(xb))$.

This means that in \mathcal{A} we will have $\delta(p_0, ux) = p_1$ and $\delta(p_0, u'x) = p_2$ on one hand, and $\delta(p_1, b) = \delta(p_2, b) = p$ on the other hand (see Fig. 7).

In τ_{k+1} we would have the configuration depicted in Fig. 8.

As $|x| = k$, \mathcal{A} would not be k -reversible. \square

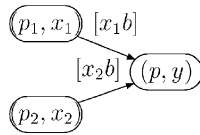


Fig. 5.

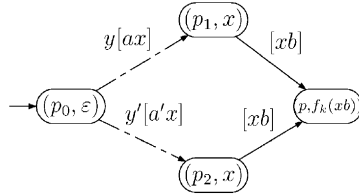


Fig. 6.

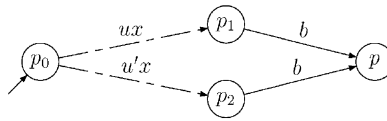


Fig. 7.

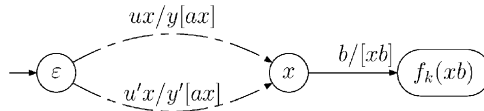


Fig. 8.

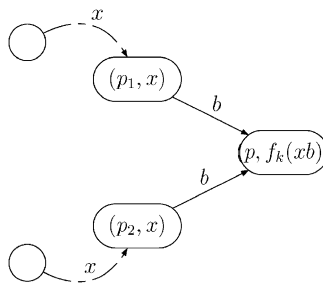


Fig. 9.

As a consequence, if L belongs to Rev_k , then L is in $\mathcal{L}_{Rev_0 * LI_{k+1}}(A^*)$. For the converse we have the following:

Proposition 14. *If $L \in \mathcal{L}_{Rev_0 * LI_{k+1}}(A^*)$ then $L \in Rev_k$.*

Proof. Let $L \in \mathcal{L}_{Rev_0 * LI_{k+1}}(A^*)$. Then L is recognized by the wreath product $M \circ N$ where $M \in \mathbf{Rev}_0$ and $N \in \mathbf{LI}_{k+1}$. We have to prove that if $Y = L(\mathcal{B})$, with $\mathcal{B} = (P, B, \delta, p_0, F)$ and $B \subseteq A^{k+1}$ is such that $Y \in Rev_0$ and if $L = L(\mathcal{B} \circ \tau_{k+1})$, then $L \in Rev_k$.

By contradiction, if $L \notin Rev_k$ there will be in $\mathcal{B} \circ \tau_{k+1}$ the configuration depicted in Fig. 9, with $|x| = k$.

Then, in \mathcal{B} we will have $\delta(p_1, [xb]) = \delta(p_2, [xb]) = p$, which means that $L \notin Rev_0$. \square

As a consequence of the two former propositions, we have the following:

Proposition 15. $Rev_k = \mathbf{Rev}_0 * \mathbf{LI}_{k+1}$.

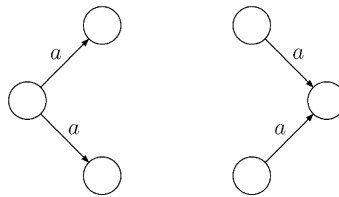


Fig. 10. Configurations that violate zero-reversibility condition in automata.

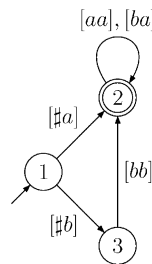


Fig. 11. Zero-reversible automaton obtained with the transduction of $S = \{a, aa, bb, bbaa\}$.

4.2.2. Example of inference

We are going to apply our method to the inference of reversible languages in the sense of Angluin. An algorithm that, given a positive sample S , outputs an automaton that recognizes the smallest R_0 language that contains S has been proposed in [3]. The family of reversible languages in the sense of Angluin is not a pseudovariety of languages but, as every reversible language in the sense of Angluin is reversible in the sense of Pin, the wreath product principle for ordered semigroups [30] allows us to use the proposed scheme in this case too.

We briefly recall Angluin’s algorithm for zero-reversible languages. From a positive sample S it outputs a zero-reversible automaton that recognizes the smallest zero-reversible language that contains S . It is a merging states algorithm that starts building the prefix tree acceptor for S and updating a list of pairs of states to be merged. This list is initialized with pairs (q, q') , where $q \in F$ and $q' \in F - \{q\}$. Every time a merge is performed it has to keep track of further merges implied by this one. This process is repeated until the list becomes empty (i.e. no further merges can be done). For details about implementation and correctness see [3].

Algorithm zero-reversible

/*Input a nonempty positive sample S */

/*Output a zero-reversible acceptor*/

Let $\mathcal{A}_0 = (Q, A, \delta, q_0, F)$ be the prefix tree acceptor for S

Let $q \in F$. **For every** $q' \in F$, add (q, q') to $LIST$ **End For.**

While $LIST \neq \emptyset$

Let (q, q') be the First element of $LIST$

Merge q and q' .

Delete (q, q') from $LIST$.

For every (q_1, q_2) that violates the zero-reversibility condition (see Fig. 10)

Add (q_1, q_2) to $LIST$.

End For.

End While

End

Using Angluin’s algorithm with the sample $S = \{a, aa, bb, bbaa\}$ we obtain the automaton of Fig. 11. The way this algorithm works makes that the output automaton is reversible in the sense of Angluin.

Multiplying this automaton by the transducer of Fig. 1 for $k = 2$ and placing the final states we obtain the automaton of Fig. 12, which recognizes the smallest 1-reversible language that contains the sample S .

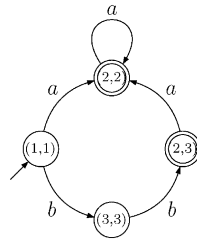


Fig. 12. 1-Reversible automaton obtained with $S = \{a, aa, bb, bbaa\}$ using cascade product.

4.3. Smallest right- \mathbf{B}_1 language that contains a positive sample S

As $\mathbf{LR} = \mathbf{R} * \mathbf{LI}$, we can infer the languages whose monoids belong to \mathbf{LR} using the same scheme as above. The base case is now the smallest \mathbf{R} -trivial (for the value k) language that contains the sample. Note that as both functions τ_k and τ_k^{-1} are bijective, the inferred language is the smallest language belonging to the family \mathbf{LR} that contains the sample. Languages belonging to this later variety of languages are generated by the congruence in which two words are equivalent if they have the same set of subwords of length k of segments of length m for given values of k and m , considering the order of appearance of those subwords.

Note that if the language belongs to the family \mathbf{B}_1 the base case will be a piecewise testable language and will be eventually inferred using this scheme.

We briefly recall the k -PWTI algorithm that obtains the smallest language whose monoid is R -Trivial that contains a sample S . If the target language is piecewise testable and we have a characteristic sample (i.e. a word for every possible order of appearance of the subwords in the words), this language and will also be inferred. For details about implementation and correctness see [13].

The algorithm relies on a combinatorial property of the words of this type of languages discovered by Simon [34] that states that if $u \in A^*$ and $a \in A$, ua has the same set of subwords of length k than u if $u = u_1u_2 \dots u_k$ and $\alpha(u_1) \supseteq \dots \supseteq \alpha(u_k) \ni a$.

Algorithm k -Piecewise Testable Inference (k -PWTI)

Input: $k \in \mathbf{N}$, S set of words over A^*

Output DFA $\mathcal{A} = (Q, A, \delta, q_1, F)$, consistent with S such that $L(\mathcal{A})$ is the smallest k -right Piecewise testable language that contains S

Method:

$\mathcal{A} = (\{q_1\}, A, \emptyset, q_1, \emptyset)$

For every word $x = a_1 \dots a_n \in S$

Try to recognize x by \mathcal{A}

If $\delta(q_i, a_i) = q_{i+1}$, $i < m < n$ and $\nexists \delta(q_m, a_m)$

Then

Add a new state q to Q

$\delta(q_m, a_m) = q$

If $m = n$

Add q to F

End If

For every $a \in A$

If $a_1 \dots a_m$ and $a_1 \dots a_m a$ have the same set of k -subwords

Set $\delta(q, a) = q$

End If

End For

End If

End

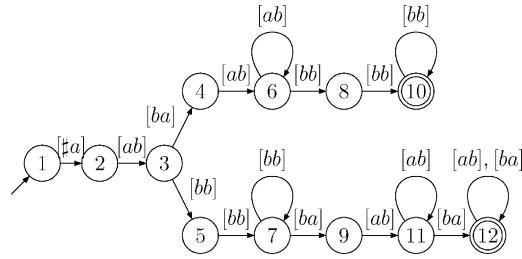


Fig. 13. Base case for the sample $\{ababbb, abbbabab\}$ for inference of languages of type right B_1 for $k = 2$ and $m = 2$.

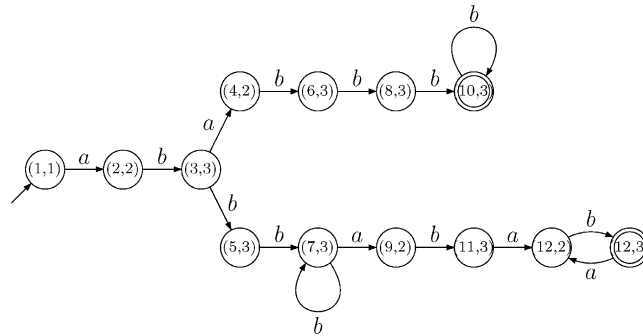


Fig. 14. Automaton that recognizes the smallest $(2, 2)$ -right B_1 language for the sample $\{ababbb, abbbabab\}$.

For example in Fig. 13 an automaton that recognizes the smallest \mathbf{R} -trivial (for the value $k = 2$) language that contains $\{ababbb, abbbabab\}$ is obtained using algorithm k -PWTL.

The product of this automaton by the transducer of Fig. 1 for $k = 2$ gives the automaton of Fig. 14, which recognizes the smallest right B_1 language (for values $k = 2$ and $m = 2$) that contains the sample $\{ababbb, abbbabab\}$.

4.4. Smallest k -testable modulo p language that contains a positive sample S

The congruence that defines the family of k -testable modulo p language for $k = 1$ is reduced to calculate the number of occurrences modulo p of every symbol of a word, so for inference of the base case of the k -testable modulo p language that contains a positive sample S we can use the following algorithm:

Algorithm 1-Testable modulo p Language Inference $((1, p) - TI)$

Input: $p > 1$ S set of words over B^*

Output DFA $\mathcal{A} = (Q, B, \delta, q_0, F)$, consistent with S such that $L(\mathcal{A})$ is the smallest 1-Testable language modulo p that contains S

Method:

$B = \alpha(S) = \{a_1, a_2, \dots, a_r\}$

$Q = \{(n_1, n_2, \dots, n_r) : n_i \in \{0, \dots, p - 1\}\}$

$q_0 = (0, \dots, 0)$

For every $(n_1, n_2, \dots, n_r) \in Q$

For every $a_i \in B$

$\delta((n_1, \dots, n_r), a_i) = (n_1, \dots, n_i + 1 \bmod p, \dots, n_r)$

End For

End For

$F = \bigcup_{w \in S} \{(|w|_{a_1} \bmod p, \dots, |w|_{a_r} \bmod p)\}$

End

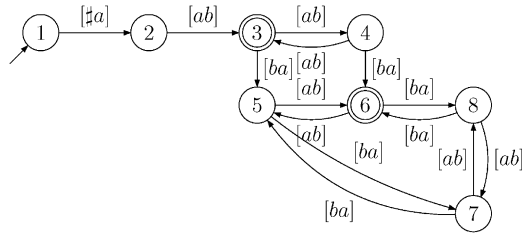


Fig. 15. Base case for the transduction of the sample $\{ababb, ab\}$ for inference of languages k -testable modulo p languages.

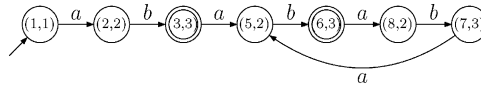


Fig. 16. Automaton that recognizes the smallest 2-testable modulo 2 language for the sample $\{abab, ab\}$.

This algorithm considers non-appearing symbols the same as if the number of appearances is a multiple of p . As this family does not seem an appropriate model for learning purposes we consider a very similar family which has also been proposed in [21,11]. The only difference with the former family is that in the new one, if a symbol does not appear in the sample, it is not considered. The corresponding variety of semigroups is the variety of locally p -idempotent and commutative semigroups, that is, the variety such that the local subsemigroups fulfill the equations $s^{p+1} = s$ and $st = ts$.

The automaton that recognizes the base case for this later family is $\mathcal{A} = (Q, B, \delta, q_0, F)$, where

- $B = \alpha(S) = \{a_1, a_2, \dots, a_r\}$,
- $Q = \{(n_1, n_2, \dots, n_r) : n_i \in \{0, \dots, p - 1\}\} \cup \{\uparrow\}$,
- $q_0 = (\uparrow, \dots, \uparrow)$ and the transition function
- $\delta((n_1, n_2, \dots, n_r), a_i) = \begin{cases} (n_1, \dots, n_i + 1 \bmod p, \dots, n_r) & \text{if } 0 \leq n_i \leq p - 1, \\ (n_1, \dots, n_{i-1}, 1, \dots, n_r) & \text{if } n_i = \uparrow. \end{cases}$

The order of appearance of symbols in the words of the sample may also be considered to avoid non-desired over-generalizations. To do so, we have to characterize the states of the automaton by a vector of integers modulo p and by a list of the order of appearance of every first symbol.

Applying these considerations to the sample $S = \{abab, ab\}$ we obtain the automata of Fig. 15. Note that the symbol $\#[a]$ is only used at the beginning, so it is not considered after.

If we now realize the cascade product with the transducer for $k = 2$, we obtain the automata of Fig. 16, that recognizes the smallest 2-testable modulo 2 language that contains S .

4.5. Smallest k -testable language in the strict sense that contains a positive sample S

The family of Locally Testable Languages (*LTSS*) in the strict sense has been defined by McNaughton and Papert [25]. We recall that, given a value of $k > 0$, a language L is k -Testable in the strict sense (k -*TSS*) if there exist three sets $I, F \subseteq A^{k-1}$ and $T \subseteq A^k$ such that $L \cap A^{k-1}A^* = IA^* \cap A^*F - A^*TA^*$. *LT* languages are the boolean closure of *LTSS* languages. Local languages, also called N -grams, have proved to be useful in pattern recognition. An algorithm that infers the family of k -*TTSS* in linear time has been proposed in [15].

Using the same scheme as before we can learn the smallest k -testable language in the strict sense that contains a sample S . For $k = 1$ and a given sample S , the smallest locally testable language in the strict sense that contains S is $\alpha(S)^*$, as in the former expression, $I = F = \{\varepsilon\}$ and T is the set of symbols not present in S . Then the automaton that recognizes the smallest 1-testable language in the strict sense that contains the transduction of the sample is $\mathcal{A} = (\{q_0\}, t_k(S), \delta_2, q_0, \{q_0\})$, where $\delta_2(q_0, a) = q_0$, for every $a \in t_k(S)$. The automata for the cases $k = 2$ and 3 and the sample $S = \{a, aba, abbaba\}$ are shown in Fig. 17.

If we realize the cascade product of each of these automata by the sequential transducers of Fig. 1 we obtain the output automata shown in Fig. 18.

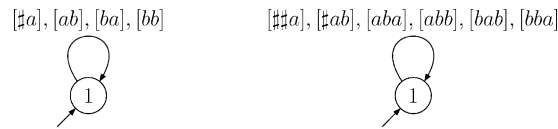


Fig. 17. Base cases for the inference of the smallest k -testable language in the strict sense that contains the sample $S = \{a, aba, abbaba\}$ for the values $k = 2$ and 3.

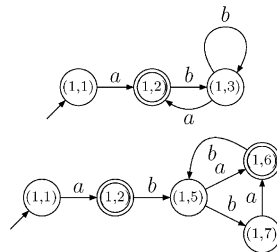


Fig. 18. Automaton that recognizes the smallest k -testable languages in the strict sense for the sample $S = \{a, aba, abbaba\}$ and the values $k = 2$ (top) and 3 (bottom).

5. Conclusions

We have presented a general inference method for languages that belong to varieties of the form $\mathbf{V} * \mathbf{LI}$ using tools that had been developed in semigroup theory. The method unifies existing algorithms for inference of some of those families (locally testable, reversible) and can be applied to some others that had not been inferred yet (B1, modular locally testable). This method might be used in every variety of the form $\mathbf{V} * \mathbf{LI}$ as soon as we have characterized the languages belonging to it.

References

- [2] D. Angluin, Inductive inference of formal languages from positive data, *Inform and Control* (1980) 117–135.
- [3] D. Angluin, Inference of reversible languages, *J. ACM* 29 (3) (1982) 741–765.
- [4] J. Berstel, *Transductions and Context Free Languages*, Teubner, Stuttgart, 1979.
- [6] J.A. Brzozowski, Languages of R-trivial monoids, *J. Comput. System Sci.* 20 (1980) 32–49.
- [7] J.A. Brzozowski, I. Simon, Characterizations of locally testable events, *Discrete Math.* 4 (1973) 243–271.
- [8] S. Eilenberg, *Automata, Languages and Machines*, Vol. A and B, Academic Press, New York, 1976.
- [11] P. García, *Modular Locally Testable Languages*, Unpublished Manuscript, 1998.
- [12] P. García, J. Ruiz, Right and left locally testable languages, *Theoret. Comput. Sci.* 246 (1–2) (2000) 253–264.
- [13] P. García, J. Ruiz, Learning k -testable and k -piecewise testable languages from positive data, *Grammars* 7 (2004) 125–140.
- [15] P. García, E. Vidal, Inference of k -testable languages in the strict sense and applications to syntactic pattern recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* 12/9 (1990) 920–925.
- [16] E.M. Gold, Language identification in the limit, *Inform. and Control* 10 (1967) 447–474.
- [20] J. Hopcroft, J. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading, MA, 1979.
- [21] R. Knast, *Periodic and strictly periodic local testability*, Technical Report, Institute of Mathematics, Polish Academy of Sciences, Poznam, Poland, 1976.
- [22] S. Kobayashi, T. Yokomory, Learning approximately regular languages with reversible languages, *Theoret. Comput. Sci.* 174 (1997) 251–257.
- [25] R. McNaughton, S. Papert, *Counter-Free Automata*, MIT Press, Cambridge, MA, 1971.
- [26] J.E. Pin, *Varieties of Formal Languages*, Plenum Press, New York, 1986.
- [27] J.E. Pin, On Reversible languages, *Proc. First LATIN Conf.*, Lecture Notes in Computer Science, Vol. 583, Springer, New York, 1992, pp. 401–416.
- [28] J.E. Pin, A variety theorem without complementation, *Russian Math.* 39 (1995) 80–90.
- [29] J.E. Pin, Personal communication, 2003.
- [30] J.E. Pin, P. Weil, The wreath product principle for ordered semigroups, *Comm. Algebra* 30 (2002) 5677–5713.
- [31] V. Radhakrisnan, G. Nagaraja, Inference of regular grammars via skeletons, *IEEE Trans. Systems Man Cybernet.* SCM-17 (1987) 982–992.
- [33] I. Simon, *Hierarchies of events dot-depth one*, Ph.D. Thesis, University of Waterloo, 1972.

- [34] I. Simon, Piecewise testable events, Lecture Notes in Computer Science, Vol. 33, Springer, New York, 1980.
- [35] H. Straubing, Finite semigroup varieties of the form V^*D , J. Pure Appl. Algebra 36 (1985) 52–94.
- [36] H. Straubing, Finite Automata, Formal Logic and Circuit Complexity, Birkhäuser, Basel, 1994.
- [37] Y. Zalcstein, Locally testable languages, J. Comput. System Sci. 6 (1972) 151–167.