

# Finite Automata and Unions of Regular Patterns with Bounded Constant Segments

Antonio Cano and Pedro García

Departamento de Sistemas Informáticos y Computación,  
Universidad Politécnica de Valencia, Valencia, Spain  
{acano, pgarcia}@dsic.upv.es

**Abstract.** The class of unbounded unions of regular pattern languages with bounded constant segments is identifiable from positive data in the limit [1]. Otherwise, no efficient algorithm that performs the inference of this class of languages is known. We propose a solution to this problem using the existing connexion between the positive variety of languages of dot depth  $1/2$ ,  $\mathcal{LJ}^+$  [2] and the class of unbounded union of pattern languages  $\mathcal{RP}^+\mathcal{L}$ .

## 1 Introduction

Pattern languages have been introduced by Angluin in [3], where she has shown that they are identifiable from positive data. They have been used for machine discovery of protein motifs from amino acid sequences in [4].

A pattern is a word on the alphabet  $\Sigma \cup X$ , where  $\Sigma$  is a finite alphabet (of constant symbols) and  $X$  is a disjoint countable alphabet (of variables). Given a pattern  $p$ , the language  $L(p)$  is defined as the set of words on  $\Sigma$  obtained by replacing the variables in  $p$  by constant words.

The class of pattern languages is efficiently identifiable from positive data in the limit [3]. From the point of view of applications, the use of only one pattern is not enough flexible and it would be more interesting to use several patterns. The problem for doing so is that the class of pattern languages is not closed under union. If we consider the closure under union of the class of pattern languages  $\mathcal{P}^+\mathcal{L}$  we obtain a class (unbounded unions of pattern languages) that is not identifiable from positive samples, as any word  $w \in \Sigma^*$  is also a pattern such that  $L(w) = \{w\}$  and, consequently, the class of unions of pattern languages contains every finite language and also some infinite languages. The same problem still remains when we consider the union of regular pattern languages  $\mathcal{RP}^+\mathcal{L}$ , that is, patterns in which any variable occurs at most once in the pattern. Shinohara and Arimura [1] have considered an interesting restriction of the class of unbounded unions of regular pattern languages, for any  $k > 0$  they consider the class of unions of pattern in which the length of the constant segments is bounded by  $k$  (unbounded union of regular pattern languages with bounded constant segments ( $\mathcal{RP}_k^+\mathcal{L}$ )). In other words, a language belongs to  $\mathcal{RP}^+\mathcal{L}$  if and only if it belongs to  $\mathcal{RP}_k^+\mathcal{L}$  for some  $k > 0$ . For any  $k$ ,  $\mathcal{RP}_k^+$  is identifiable from positive data in the

limit [1]. The proof given by Shinohara and Arimura uses a theorem of Higman on well-quasi ordering. Nevertheless, they do not give an efficient algorithm to identify any class  $\mathcal{RP}_k^+\mathcal{L}$ .

In this work we propose an efficient algorithm for the inference of  $\mathcal{RP}_k^+\mathcal{L}$  using a relation between  $\mathcal{RP}_k^+\mathcal{L}$  and the positive variety of languages  $\mathcal{LJ}^+$  [2], also known as languages of dot-depth 1/2.

We show here that, for any  $k > 0$ ,  $\mathcal{RP}_k^+\mathcal{L} \subseteq \mathcal{LJ}_{k+1}^+$ . From this fact and from the relation  $\mathbf{LJ}_k^+ = \mathbf{J}^+ * \mathbf{LI}_k$  [5] the problem of the inference of  $\mathcal{RP}_k^+\mathcal{L}$  from positive data can be solved in an easy way through the scheme of inference of languages in varieties of the form  $\mathbf{V} * \mathbf{LI}$  proposed in [6] and [7]. So, we give an efficient algorithm to learn languages from  $\mathcal{RP}_k^+\mathcal{L}$ .

## 2 Preliminaries

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].

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

Given  $x \in \Sigma^*$ , if  $x = uvw$  with  $u, v, w \in \Sigma^*$ , then  $u$  (resp.  $w$ ) is called *prefix* (resp. *suffix*) of  $x$ , whereas  $v$  is called a *segment* of  $x$ . The set of prefixes (resp. suffixes) of a word  $x$  will be denoted as  $Pr(x)$  (resp.  $Suf(x)$ ). We will also denote by  $Pr_k(x)$  the prefix of length  $k$  of  $x$  (resp. by  $Suf_k(x)$  the suffix of length  $k$  of  $x$ ). Given  $x, y \in \Sigma^*$ , we say that  $x = a_1a_2 \cdots a_n$ , with  $a_i \in \Sigma$ ,  $i = 1, 2, \dots, n$  is a *subword* of  $y$ , and we denote this relationship by  $x | y$  if  $y = z_0a_1z_1a_2 \cdots a_nz_n$ , with  $z_i \in \Sigma^*$  for  $i = 0, 1, \dots, n$ .

A Nondeterministic Finite Automaton (*NFA*) is defined as a quintuple  $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$  where  $Q$  is a finite set of states,  $\Sigma$  is a finite alphabet,  $Q_0 \subseteq Q$  is the set of initial states,  $F \subseteq Q$  is the set of final states and  $\delta$  is a partial function from  $Q \times \Sigma$  into  $\mathcal{P}(Q)$ , which can be extended to a function from  $\mathcal{P}(Q) \times \Sigma$  into  $\mathcal{P}(Q)$  by establishing  $\delta(Q', a) = \bigcup_{q \in Q'} \delta(q, a)$  for any  $Q' \subseteq Q$  and  $a \in \Sigma$ . It can also be extended to a function from  $\mathcal{P}(Q) \times \Sigma^*$  into  $\mathcal{P}(Q)$  by establishing  $\delta(Q', \epsilon) = Q'$  and  $\delta(Q', xa) = \delta(\delta(Q', x), a)$ , for every  $Q' \subseteq Q$ ,  $x \in \Sigma^*$  and  $a \in \Sigma$ . If in the previous definition we take  $Q_0 = \{q_0\}$  with  $q_0 \in Q$  and  $\delta$  as a function from  $Q \times \Sigma^*$  into  $Q$ , we obtain the definition of Deterministic Finite Automaton (*DFA*).

A word  $x$  is accepted by an automaton  $\mathcal{A}$  if  $\delta((Q_0), x) \cap F \neq \emptyset$ . The set of words accepted by  $\mathcal{A}$  is denoted by  $L(\mathcal{A})$ .

A sequential machine is a sextuple  $\mathcal{A} = (Q, \Sigma, \Delta, \delta, \lambda, F)$  where  $Q$ ,  $\Sigma$  and  $F$  are defined in the same way as in a *DFA*,  $\Delta$  is the output alphabet and the output function  $\lambda$  is a function that maps  $Q \times \Sigma$  into  $\Delta^*$ , which can be extended

to  $Q \times \Sigma^*$  by establishing  $\lambda(q, \epsilon) = \epsilon$ , and  $\lambda(q, xa) = \lambda(q, x)\lambda(\delta(q, x), a)$ , for every  $q \in Q$ ,  $x \in \Sigma^*$  and  $a \in \Sigma$ .

We use the model of learning called identification in the limit [9]. An algorithm  $A$  identifies a class of languages  $H$  *in the limit* if and only if for any  $L \in H$ , on input of any presentation of  $L$ , the infinite sequence of output languages obtained by  $A$  converges to  $L$ .

### 2.1 Formal Languages

A language is of level 1/2 in the Straubing-Thérien hierarchy if it is a finite union of languages of the form  $\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \cdots \Sigma^* a_n \Sigma^*$ , where  $a_1, \dots, a_n \in \Sigma$ . The family of languages of level 1/2 in the Straubing-Thérien's hierarchy forms the positive variety  $\mathcal{J}^+$  corresponding to the variety of ordered monoids  $\mathbf{J}^+$  [5].

The languages of dot-depth 1/2 are finite unions of languages of the form  $u_0 \Sigma^* u_1 \Sigma^* \cdots u_{k-1} \Sigma^* u_k$ , where  $k \geq 0$  and  $u_0, \dots, u_k \in \Sigma^*$ . The family of languages of dot-depth 1/2 forms the positive variety  $\mathcal{LJ}^+$  corresponding to the variety of ordered semigroups  $\mathbf{LJ}^+$  [5].

The next theoretical result comes from [5] and characterizes some subclasses of  $\mathcal{LJ}^+$ .

Given some words  $u_1, u_2, \dots, u_n$  of the same length, we define

$$L(u_1, \dots, u_n) = \{u \in \Sigma^+ \mid u_1, \dots, u_n \text{ occur in this order as segments of } u\}$$

If we set  $u = u_1 u_2 \cdots u_n \in (\Sigma^k)^*$  with  $u_i \in \Sigma^k$  for  $i \in \{1, \dots, n\}$ , we also denote  $L(u_1, \dots, u_n)$  by  $L(u)$ .

**Theorem 1.** [5] *Let  $L$  be a language of  $\Sigma^+$ . The following conditions are equivalent.*

- (1)  $L$  is of dot-depth 1/2,
- (2)  $L$  is a finite union of languages of the form  $\{u\}$ , with  $|u| < k - 1$  or  $p \Sigma^* \cap L(u_1, \dots, u_n) \cap \Sigma^* s$ , where, for some  $k > 1$ ,  $p, s \in \Sigma^{k-1}$  and  $u_1, \dots, u_n$  is a sequence of words of  $\Sigma^k$ .

For a given  $k > 0$ , we will denote by  $\mathcal{LJ}_k^+$  the languages defined in (2).

It is known that  $\mathbf{LJ}^+ = \mathbf{J}^+ * \mathbf{LI}$  and that  $\mathbf{LJ}_k^+ = \mathbf{J}^+ * \mathbf{LI}_k$  [5].

$\mathbf{LI}$  is the variety of locally trivial finite semigroups. For any  $k > 0$  the variety of languages corresponding to  $\mathbf{LI}_k$  consists of languages of the form  $X \Sigma^* Y \cup Z$  with  $X, Y \subseteq \Sigma^k$  and  $Z \subseteq \Sigma^{<k}$ , some other definition can be found in [10].

### 2.2 Pattern Languages

Let  $\Sigma$  be a set of constant symbols containing at least two symbols, and  $X$  be a countable set of variable symbols. We assume that  $\Sigma \cap X = \emptyset$ . A pattern  $p$  is a word on  $(\Sigma \cup X)^*$ . Note that we consider the empty word  $\epsilon$ . We denote by  $\mathcal{P}$  the set of all patterns. The length of a pattern  $p \in \mathcal{P}$ , will be denoted by  $|p|$ . A substitution  $\rho$  is a homomorphism from patterns to patterns that maps every constant to itself. For a pattern  $p$  and a substitution  $\rho$ . We say that a pattern

$q$  is a *generalization* of  $p$ , or  $p$  is an instance of  $q$ , and we denote that fact by  $p \preceq q$ , if there is a substitution  $\rho$  such that  $\rho(q) = p$ .

The language defined by a pattern  $p \in \mathcal{P}$  is the set  $L(p) = \{w \in \Sigma^* \mid w \preceq p\}$ . We denote by  $\mathcal{PL}$  the class of all pattern languages.

In this paper, we are specially concerned about a subclass of  $\mathcal{P}$ . A pattern  $p \in \mathcal{P}$  is regular, if each variable appears at most once in  $p$ , i. e. for any  $x \in X$ , the number of occurrences of  $x$  in  $p$   $|p|_x \leq 1$ . A *regular pattern language* is a pattern language defined by a regular pattern. We denote by  $\mathcal{RP}$  the set of all regular patterns, and by  $\mathcal{RPL}$  the set of all regular patterns languages.

We are also concerned with unions of languages defined by patterns. By  $\mathcal{P}^+$  we denote the class of all nonempty finite subsets of  $\mathcal{P}$ . Analogously, by  $\mathcal{RP}^+$  we denote the class of all nonempty finite subsets of  $\mathcal{RP}$ , and by  $\mathcal{RPL}^+$  the corresponding class of languages.

The next proposition shows that the class of unions of pattern languages is exactly the positive variety  $\mathcal{LJ}^+$ .

**Proposition 1.** *The class of unions of regular pattern languages  $\mathcal{RPL}^+$  is the positive variety of languages  $\mathcal{LJ}^+$ .*

*Proof.* By the definition of  $\mathcal{LJ}^+$ , it suffices to see that for any pattern  $p$ ,  $L(p) = u_0 \Sigma^* u_1 \Sigma^* \cdots u_{k-1} \Sigma^* u_k$ , for some  $k \geq 0$  and  $u_0, \dots, u_k \in \Sigma^*$ , and that for any language  $L$  in this form, there exists a pattern  $p$ , such that  $L(p) = L$ .  $\square$

Finally we are interested in unions of bounded pattern languages. Given an integer  $k \geq 0$  a  $k$ -bounded pattern is a pattern that has at most  $k$  constant consecutive symbols. We will denote the set of all these patterns as  $\mathcal{P}_k$ , and as in the previous cases we will denote by  $\mathcal{RP}_k$  and  $\mathcal{RPL}_k$ , the sets of  $k$ -bounded regular patterns and  $k$ -bounded regular pattern languages. From the definition of bounded pattern we obtain the definition of union of  $k$ -bounded regular pattern and  $k$ -bounded regular pattern languages, that will be denoted by  $\mathcal{RP}_k^+$  and  $\mathcal{RPL}_k^+$  respectively.

### 3 Inferring $\mathcal{J}^+$

In this section we describe an algorithm for the inference of languages that belongs to  $\mathcal{J}^+$ .

Given a sample  $S = \{x_1, x_2, \dots, x_n\}$  we can associate a language  $L_{\mathcal{J}^+S} \in \mathcal{J}^+$  as follows

$$L_{\mathcal{J}^+S} = \{x \in \Sigma^* \mid \text{there exists } i \in \{1, \dots, n\} \text{ such that } x_i \mid x\}.$$

**Proposition 2.**  *$L_{\mathcal{J}^+S}$  is the smallest language in  $\mathcal{J}^+$  that contains  $S$ .*

*Proof.* If  $K \in \mathcal{J}^+$  and  $w \in K$ , any word  $x$  such that  $w \mid x$  belongs to  $K$ . If furthermore  $S \subseteq K$ ,  $K$  contains all words  $x$  such that  $x_i \mid x$  for some  $x_i \in S$ . And then,  $L_{\mathcal{J}^+S} \subseteq K$ .  $\square$

Algorithm 1 yields a *NFA* by constructing for every word  $x_i$  an automaton that accepts all words which have  $x_i$  as subwords, we order  $S$  and check the

acceptance of any word before constructing the automaton in order to save automaton size.

For the study of the convergence of Algorithm 1 we have that for any alphabet  $\Sigma$  and any language  $L \in \mathcal{J}^+(\Sigma^*)$ ,  $L$  can be written as  $\bigcup_{1 \leq i \leq m} \Sigma^* a_{i,1} \Sigma^* \dots \Sigma^* a_{i,n_i} \Sigma^*$ . Then, if we use as input of the algorithm the set  $S = \{a_{1,1} \dots a_{1,n_1}, \dots, a_{m,1} \dots a_{m,n_m}\}$ , we have that  $L_{\mathcal{J}^+ S} = L$ , and the result follows from Proposition 2.

The time complexity of Algorithm 1 is  $N \cdot \Sigma$ . Let  $S = \{x_1, x_2, \dots, x_n\}$  and let  $N = |x_1| + |x_2| + \dots + |x_n|$ . As for any  $i \in \{1, \dots, n\}$  The  $\mathcal{J}^+$  Inference Algorithm constructs an automaton whose number of states is  $|x_i| + 1$ , having every state at most  $|\Sigma|$  transitions, the complexity of the algorithm is  $N \cdot \Sigma$ . Nevertheless, if we consider an implementation where the construction of transitions is linear, the overall algorithm complexity would be linear with the size of the input data.

---

**Algorithm 1.**  $\mathcal{J}^+$  Inference

**Input:**  $S$  set of words over  $\Sigma^*$

**Output:** NFA  $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ , such that  $L(\mathcal{A}) = L_{\mathcal{J}^+ S}$

**Method:**

$Q = \emptyset$ ;  $\delta = \emptyset$ ;  $F = \emptyset$ ;  $Q_0 = \emptyset$

**Order  $S$  by decreasing length as:**  $S = \{x_1, x_2, \dots, x_m\}$

**For**  $x_i = a_1 a_2 \dots a_n \in S$  **Do**

**If**  $\delta(Q_0, x) \cap F = \emptyset$  **Then**

$Q = Q \cup \{(i, 0), (i, 1), \dots, (i, n)\}$

**For**  $j = 0$  **To**  $n - 1$  **Do**

$\delta = \delta \cup ((i, j), a_j, (i, j + 1))$

**For**  $c \in \Sigma \setminus \{a_j\}$

$\delta = \delta \cup ((i, j), c, (i, j))$

**For**  $c \in \Sigma$

$\delta = \delta \cup ((i, n), c, (i, n))$

$Q_0 = Q_0 \cup \{(i, 0)\}$

$F = F \cup \{(i, n)\}$

**Return**  $(\mathcal{A})$

---

## 4 Inferring $\mathcal{LJ}^+$

We recall the following definitions and theorems that will lead us in the inference process. We follow a similar scheme to the scheme used in [6, 7].

**Theorem 2 (Ginzburg and Rose, see [11]).** *Let  $\tau : \Sigma^* \rightarrow \Gamma^*$  be the sequential function realized by the transducer  $\tau = (P, \Sigma, \Gamma, \delta_1, \lambda, p_0, F)$  and let  $\mathcal{A} = (Q, \Gamma, \delta_2, q_0, F')$  be an automaton such that  $L = L(\mathcal{A})$ . The language  $\tau^{-1}(L) \subseteq \Sigma^*$  is recognized by the cascade product  $\mathcal{A} \circ \tau = (Q \times P, \Sigma, \delta, [q_0, p_0], F' \times F)$ , with the transition function defined as  $\delta([q, p], a) = (\delta_2(q, \lambda(p, a)), \delta_1(p, a))$ .*

We define now the transduction  $\tau_{k,F}$ , for  $k$  and  $F \subseteq \Sigma^{k-1}$ , that will be used in the sequel.

**Definition 1.** For  $k > 0$  and a finite set of word  $F \subseteq \Sigma^{k-1}$ . Let  $\tau_{k,F} = (Q, \Sigma, B, \delta, \lambda, q_0, F \cup \Sigma^{<k-1})$  be a sequential machine defined as  $Q = \cup_{i=0}^{k-1} \Sigma^i$ ,  $p_0 = \varepsilon$ ,  $B = \cup_{i=1}^{k-1} \#^{k-i} \Sigma^i \cup \Sigma^k$  and for every  $p \in Q$  and  $a \in A$ , the transition and output functions are respectively defined as:

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

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

The sequential machine  $\tau_{k,F}$ , for a given  $k > 0$  and a word  $x \in \Sigma^*$ , outputs a word  $\tau_{k,F}(x)$  whose symbols are the segments of length  $k$  (considering as initial segments  $\#^{k-1}a_1 \mid \#^{k-2}a_1a_2 \mid \dots \mid \#a_1 \dots a_{k-2}$  being  $w = a_1 \dots a_n \in \text{Prefix}(x)$ ) of  $x$ , in order, and the last segment of length  $k - 1$  belong to  $F$ . Examples of  $\tau_{k,F}$  for the values  $k = 2$  with  $F = \{a\}$  and  $k = 3$  with  $F = \{aa, bb\}$  for  $\Sigma = \{a, b\}$  can be seen in Figure 1.

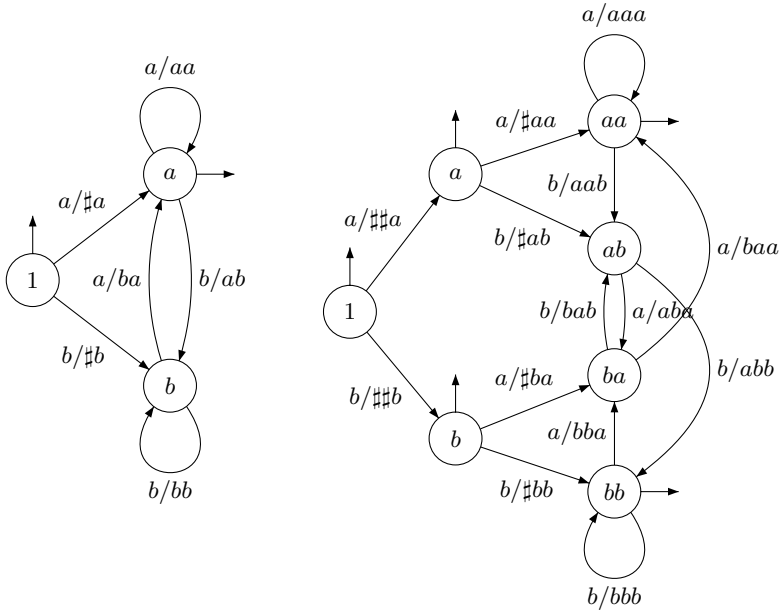


Fig. 1. Transducers  $\tau_{2,\{a\}}$  and  $\tau_{3,\{aa,bb\}}$

In order to use  $\tau_{k,F}$  in some algebraic proofs, we define the morphism  $h$  from  $B^*$  into  $(\Sigma^k)^*$  for any word  $b \in B^*$  by setting

$$h(b) = \begin{cases} b & \text{if } b \in \Sigma^k \\ 1 & \text{otherwise} \end{cases}$$

Now, we describe the algorithm 2 in order to infer languages from  $\mathcal{LJ}_k^+$ . This algorithm will use some of the above elements.

Given a word  $x$ , we can associate to this word a language  $L_{\mathcal{LJ}_k^+ \{x\}}$  in  $\mathcal{LJ}_k^+$  as follows,

$$L_{\mathcal{LJ}_k^+ \{x\}} = \begin{cases} x & \text{if } |x| < k \\ Pr_{k-1}(x)\Sigma^* \cap L(h(\tau_k(x))) \cap \Sigma^* Suf_{k-1}(x) & \text{if } |x| \geq k. \end{cases}$$

Given a sample  $S = \{x_1, x_2, \dots, x_n\}$ , we can extend the previous definition by associating the language  $L_{\mathcal{LJ}_k^+ S} \in \mathcal{LJ}^+$  as follows

$$x \in L_{\mathcal{LJ}_k^+ S} \Leftrightarrow \text{there exists } i, \text{ such that } x \in L_{\mathcal{LJ}_k^+ \{x_i\}}.$$

**Proposition 3.**  $L_{\mathcal{LJ}_k^+ S}$  is the smallest language in  $\mathcal{LJ}_k^+$  that contains  $S$ .

*Proof.* It suffices to show that the result holds for  $S = \{x\}$  for some  $x \in \Sigma^*$ . The result is trivial if  $|x| < k$ .

If  $S \in L$  and  $L \in \mathcal{LJ}_k^+$  then for any  $i \in \{1, \dots, n\}$  there exists  $y_i$  with  $\tau_{k, \Sigma^{k-1}}(y_i) \mid \tau_{k, \Sigma^{k-1}}(x_i)$  and  $Pr_{k-1}(x_i)\Sigma^* \cap L(h(\tau_{k, \Sigma^{k-1}}(y_i))) \cap \Sigma^* Suf_{k-1}(x_i) \subseteq L$ . So, for any  $i \in \{1, \dots, n\}$ ,  $L_{\mathcal{LJ}_k^+ \{x_i\}} \subseteq L$ . And then,  $L_{\mathcal{LJ}_k^+ S} \subseteq L$ .  $\square$

The algorithm 2 tries to calculate  $L_{\mathcal{LJ}_k^+ \{x_i\}}$  for any  $x_i$  in  $S$ . We order  $S$  and check the acceptance of any word before constructing the automaton. To see that the algorithm calculates  $L_{\mathcal{LJ}_k^+ \{x_i\}}$  for any  $x_i$ , it suffices to see that if  $|x_i| < k-1$ ,  $x_i$  is accepted by the definition of the cascade product and by the fact that the transduction is filled by the symbols  $\sharp$  on the left. If  $|x_i| \geq k$ , the language of the automaton contains  $L(u_1, \dots, u_n)$  by the definition of the transducer  $\tau_{k, F}$ , furthermore since the transduction is filled by the symbols  $\sharp$  on the left this implies that the prefix of all words belonging to the language accepted by the automaton is exactly  $Pr_k(x)$ . Finally, by choosing  $F = Suf_{k-1}(x)$ , the suffix of all all word belonging to the language accepted by the automaton is exactly  $Suf_{k-1}(x)$ .

---

### Algorithm 2. $\mathcal{LJ}_k^+$ Inference

**Input:**  $S$  set of words over  $\Sigma^*$  and  $k$  a positive integer

**Output:** NFA  $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ , consistent with  $S$  such that  $L(\mathcal{A}) = L_{\mathcal{LJ}_k^+ S}$

**Method:**

$Q = \emptyset$ ;  $\delta = \emptyset$ ;  $F = \emptyset$ ;  $Q_0 = \emptyset$

**Order  $S$  by decreasing length as:**  $S = \{x_1, x_2, \dots, x_m\}$

**For**  $1 \leq i \leq m$  **Do**

**If**  $\delta(Q_0, x) \cap F = \emptyset$  **Then**

$\mathcal{A}' = (Q', \Sigma', \delta', Q'_0, F') = \mathcal{J}^+ \text{Inference}(\tau_{k, Suf_{k-1}(x)}(x_i))$

$\mathcal{A} = \mathcal{A} \cup \mathcal{A}' \circ \tau_{k, Suf_{k-1}(x)}$

**Return**  $(\mathcal{A}')$

---

The convergence of Algorithm 2 is a direct consequence of Proposition 3.

The time complexity of Algorithm 2 is  $N * |\Sigma|^{k+1}$ . Let  $S = \{x_1, x_2, \dots, x_n\}$  and let  $N = |x_1| + |x_2| + \dots + |x_n|$ . The size of the transducer  $\tau_{k, F}$  is of order  $|\Sigma^k|$  but computing  $\tau_{k, F}(x_i)$  for  $i \in \{1, \dots, n\}$ , is of order  $|x_i|$ . Since the only remaining steps to be done by the algorithm are: to apply the Algorithm 2 and to calculate the cascade product, the complexity of the Algorithm 2 is  $N * |\Sigma|^{k+1}$ .

## 5 Inferring Finite Unions of Pattern Languages with Constant Segments

In this section we show that we can infer  $k$ -bounded pattern languages by using the inference algorithm used to learn  $\mathcal{LJ}^+$  languages.

In order to do so, we define a new product of automata and transducers. This product is in some sense the inverse of the cascade product.

**Definition 2.** Let  $\mathcal{A} = (Q_1, \Sigma, \delta, q_0, F)$  be a finite deterministic automaton, and let  $\tau = (Q_2, \Sigma, \Gamma, \delta', \lambda, p_0, F')$  be a sequential transducer. We define the product  $\mathcal{A}\bar{\sigma}\tau = (Q_1 \times Q_2, \Gamma, \delta'', (q_0, p_0), F \times F')$  with  $\delta''((q, p), b) = (q', p')$  if  $\delta'(p, a) = p'$ ,  $\lambda(p, a) = b$  and  $\delta(q, a) = q'$ .

The next proposition gives us the meaning of this operation.

**Proposition 4.** Given a DFA  $\mathcal{A} = (Q_1, \Sigma, \delta, q_0, F)$  and a finite transducer  $\tau = (Q_2, \Sigma, \Gamma, \delta', \lambda, p_0, F')$ ,  $L(\mathcal{A}\bar{\sigma}\tau) = \tau(L(\mathcal{A}))$ .

*Proof.* Let  $w \in \tau(L(\mathcal{A}))$ , then there exists  $x \in L(\mathcal{A})$  such that  $\tau(x) = w$ . Then, we have  $\delta'(p_0, a) = f'$ ,  $\lambda(p_0, x) = w$  and  $\delta(q, x) = f$ , for some  $f \in F$  and  $f' \in F'$ . And so,  $\delta''((q_0, p_0), w) = (f, f') \in F \times F'$ , that is  $w \in L(\mathcal{A}\bar{\sigma}\tau)$ .

Conversely, Let  $w \in L(\mathcal{A}\bar{\sigma}\tau)$ , this implies that  $\delta''((q_0, p_0), w) = (f, f')$  for some  $f \in F$  and  $f' \in F'$ , and there exists  $x \in \Sigma^*$  such that  $\delta'(p_0, a) = f'$ ,  $\lambda(p_0, x) = w$  and  $\delta(q, x) = f$ . And so,  $\tau(x) = w$  and  $x \in L(\mathcal{A})$ . And so,  $x \in \tau(L(\mathcal{A}))$ .  $\square$

Note that if there does not exist  $p \in P$  and  $a, b \in A$  such that  $\lambda(p, a) = \lambda(p, b)$  we have that  $\mathcal{A}\bar{\sigma}\tau$  is a deterministic automaton.

**Theorem 3.**  $\mathcal{RPL}_k^+$  is included in  $\mathcal{LJ}_{k+1}^+$ . Furthermore, any language belonging to  $\mathcal{RPL}_k^+$  can be obtained as a finite union of languages belonging to  $\mathcal{LJ}_{k+1}^+$ .

*Proof.* It suffices to prove the theorem for patterns of the form  $u_0x_1u_1 \cdots u_{n-1}x_nu_n$ , since the class of languages  $\mathcal{LJ}^+$  is closed under finite union. Since  $p \in \mathcal{RPL}_k$  we have that  $|u_i| \leq k$  for  $1 \leq i \leq n$ .

We now give a process to obtain the pattern language  $L(p)$  of the form  $u_0\Sigma^*u_1 \cdots u_{n-1}\Sigma^*u_n$  from languages belonging to  $\mathcal{LJ}^+$ .

Since  $p$  is a regular pattern we know that there exists an automaton  $\mathcal{A}_p$  such that  $L(p) = L(\mathcal{A}_p)$ .

Now let  $\tau_{k+1, \Sigma^{k-1}}$  be the transducer defined in Definition 1, then by Proposition 4 we know that  $L(\mathcal{A}\bar{\sigma}\tau_{k+1, \Sigma^k}) = \tau_{k+1, \Sigma^k}(L(\mathcal{A}))$ .

Let  $B = \bigcup_{i=1}^{k-1} \#^{k-i} \Sigma^i \cup \Sigma^k$ , and let us denote by  $h$  the morphism from  $B^*$  into  $(\Sigma^k)^*$  introduced in Definition 1.

We denote by  $P$  the set of acyclic paths of  $\mathcal{A}\bar{\sigma}\tau_{k+1, \Sigma^k}$  going from the initial state to some final state. Note that this set is finite. We claim that

$$L(p) = \bigcup_{\substack{X \in P \\ |x|=k, u_0 \in Pre(x) \\ |y|=k, u_n \in Suf(y)}} x\Sigma^* \cap L(h(X)) \cap \Sigma^*y \\ \bigcup \{x \in L(p) \mid |x| < k\}.$$

The result is clearly true for any  $w \in \Sigma^*$  with  $|w| < k$ . So, let us suppose  $|w| \geq k$ .

Let  $w \in L(p)$ , we know that  $\tau_{k+1, \Sigma^k}(w) \in L(\mathcal{A} \overline{\tau}_{k+1, \Sigma^k})$ , which means that there exists  $X \in P$  such that  $X \mid \tau_{k+1, \Sigma^k}(w)$ , and then, if we take  $x = Pr_k(w)$  and  $y = Suf_k(w)$ , we have that  $w \in x\Sigma^* \cap L(h(X)) \cap \Sigma^*y$ .

In the other direction let  $w \in x\Sigma^* \cap L(h(X)) \cap \Sigma^*y$  for some  $x, y \in \Sigma^k$  and  $X \in P$ . By Proposition 4 and the definition of  $P$ , there exists a word  $z \in \Sigma^*$  such that  $\tau_{k+1, \Sigma^k}(z) = X$  and  $z \in L(p)$ . Since  $z \in L(p)$  and  $p$  is  $k$  bounded we have that  $v_0v_1 \cdots v_n \mid h(X)$  with  $|v_i| = k$  and  $v_i = x_iu_iy_i$  for some  $x_i, y_i \in \Sigma^*$  with  $0 \leq i \leq n$ . Then  $\tau_{k+1, \Sigma^k}(z) \mid \tau_{k+1, \Sigma^k}(w)$  and  $v_0v_1 \cdots v_n \mid h(\tau_{k+1, \Sigma^k}(z))$ , and so, necessarily  $w \in L(p)$ . □

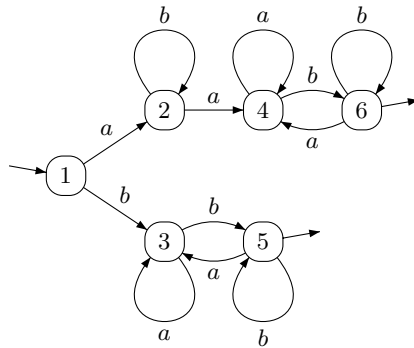
This theorem implies that  $k$ -bounded regular pattern languages can be identified by the  $\mathcal{LJ}_{k+1}^+$  Inference Algorithm in the limit.

The following example shows the behavior of the previous algorithm.

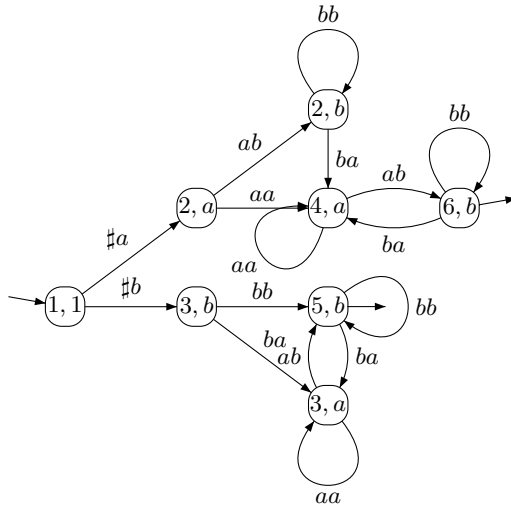
*Example 1.* Let  $p_1 = axayb$  and  $p_2 = bxb$  be 1-bounded patterns. Figure 2 shows the automaton that accepts  $L(p_1) \cup L(p_2)$ .

The automaton constructed in Theorem 3 in order to obtain the words on  $\Sigma^2$  required to learn the pattern is shown in Figure 3.

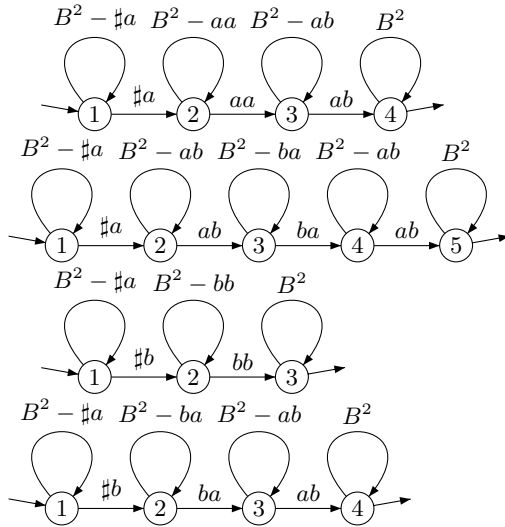
This shows that the sequences  $(aa, ab)$ ,  $(ab, ba, ab)$ ,  $(bb)$  and  $(ba, ab)$  are enough to describe  $L(p_1) \cup L(p_2)$ . By  $(aa, ab)$  we obtain  $a\Sigma^* \cap L(aa, ab) \cap \Sigma^*b$ , by  $(ab, ba, ab)$  we obtain  $a\Sigma^* \cap L(ab, ba, ab) \cap \Sigma^*b$ , by  $(bb)$  we obtain  $b\Sigma^* \cap L(bb) \cap \Sigma^*b$  and  $(ba, ab)$  we obtain  $b\Sigma^* \cap L(ba, ab) \cap \Sigma^*b$ . Note that they are the only paths without cycles from the initial to the final states.



**Fig. 2.** Automaton accepting  $L(p_1 \cup p_2)$  for the patterns  $p_1 = axayb$  and  $p_2 = bxb$



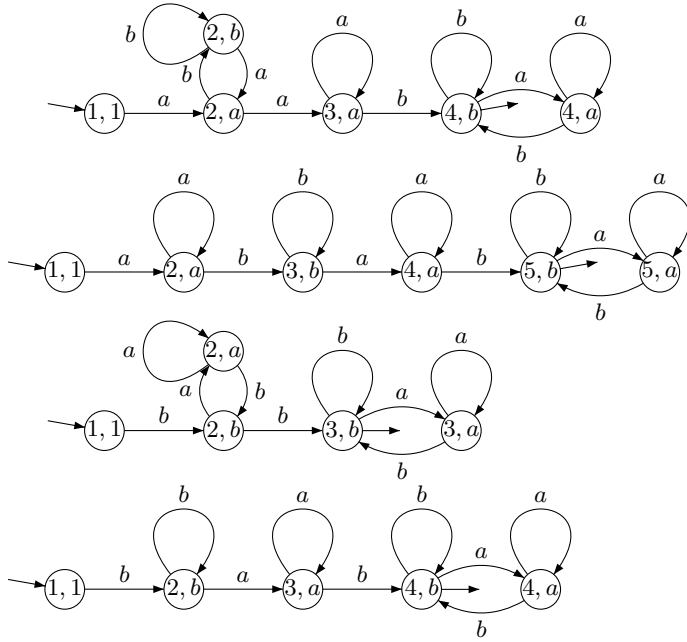
**Fig. 3.** Automaton accepting  $\mathcal{A}(L(p_1 \cup p_2)) \overline{\sigma} \tau_2$  for the patterns  $p_1 = axayb$  and  $p_2 = bxb$



**Fig. 4.** The automaton obtained for the  $\mathcal{J}^+$  inference algorithm for the samples  $\{(\#a, aa, ab), (\#a, ab, ba, ab), (\#a, ab, ba, ab, ba, ab), (\#b, bb), (\#b, bb, bb), (\#b, ba, ab)\}$

Otherwise,  $(aa, ab), (ab, ba, ab), (bb)$  and  $(ba, ab)$  also gives a characteristic set for the inference of  $L(p_1) \cup L(p_2)$ . Let us consider the set  $S = \{aab, abab, ababab, bb, bab, bbb\}$ , note that this set contains the characteristic set  $\{aab, abab, bb, bab\}$ . By applying the  $\mathcal{J}^+$  Inference Algorithm to the set  $\tau_{2, \Sigma}(S)$  we obtain the automaton shown in Figure 4.

Finally, Figure 5 shows the automaton obtained from the algorithm used to learn  $\mathcal{LJ}_2^+$ . We can verify that the minimal automaton obtained for the automa-



**Fig. 5.** The automaton obtained for the  $\mathcal{LJ}_2^+$  inference algorithm for the samples  $S = \{aab, abab, ababab, bb, bbb, bab\}$

ton shown in Figure 5 is exactly the automaton shown in Figure 2 that accepts the language  $L(p_1) \cup L(p_2)$ .

## 6 Conclusions

In this article we give an efficient inference algorithm for the positive varieties of languages  $\mathcal{J}^+$  and  $\mathcal{LJ}^+$ . The algorithm for  $\mathcal{LJ}^+$  is done by using the algorithm for  $\mathcal{J}^+$  and the cascade product.

By using this algorithm and some new and old theoretical results, we solve the open problem of given an efficient algorithm to infer unbounded unions of regular pattern languages with bounded constant segments proposed in [1]. This shows the that study of the algebraic theory of automata can give us some knowledge in order to perform new algorithms to be applied in practice.

## References

1. Shinohara, T., Arimura, H.: Inductive inference of unbounded unions of pattern languages from positive data. *Theoretical Computer Science* **241** (2000) 191–209
2. Pin, J.E., Weil, P.: Polynomial closure and unambiguous product. *Theory Comput. Systems* **30** (1997) 1–38

3. Angluin, D.: Finding common patterns to a set of strings. in Proc. 11th Ann. Symp. Theory of Computing (1979) 130–141
4. Arikawa, S., Kuhara, S., Miyano, S., Shinohara, A., Shinohara, T.: A learning algorithm for elementary formal systems and its experiments on identification of transmembrane domains. in Proc. 25th Hawaii Int. Conf. on System Science **1** (1992) 675–684
5. Pin, J.E., Weil, P.: The wreath product principle for ordered semigroups. Communications in Algebra **30** (2002) 5677–5713
6. Garcia, P., Ruiz, J.: Learning in varieties of the form  $\mathbf{V*LI}$  from positive data. To appear (-)
7. Garcia, P., Ruiz, J.: Learning  $k$ -testable and  $k$ -piecewise testable languages from positive data. Grammars. Special Issue on Grammar Induction (2004) 125–140
8. Hopcroft, J., Ullman, J.: Introduction to automata theory, languages and computation. Addison-Wesley (1980)
9. Gold, E.M.: Language identification in the limit. Information and Control (1967) 447–474
10. Pin, J.E.: Varieties of formal languages. North Oxford, London and Plenum, New-York (1986) (Traduction of Variétés de langages formels).
11. Berstel, J.: Transductions and Context Free Languages. Teubner (1979)