Learning Unions of k-Testable Languages *

Alexis Linard[1], Colin de la Higuera[2], and Frits Vaandrager[1]

[1] Institute for Computing and Information Science
Radboud University, Nijmegen, The Netherlands
{a.linard,f.vaandrager}@cs.ru.nl
[2] Laboratoire des Sciences du Numérique de Nantes
Université de Nantes, France
cdlh@univ-nantes.fr

Abstract. A classical problem in grammatical inference is to identify a language from a set of examples. In this paper, we address the problem of identifying a union of languages from examples that belong to several *different* unknown languages. Indeed, decomposing a language into smaller pieces that are easier to represent should make learning easier than aiming for a too generalized language. In particular, we consider k-testable languages in the strict sense (k-TSS). These are defined by a set of allowed prefixes, infixes (sub-strings) and suffixes that words in the language may contain. We establish a Galois connection between the lattice of all languages over alphabet Σ, and the lattice of k-TSS languages over Σ. We also define a simple metric on k-TSS languages. The Galois connection and the metric allow us to derive an efficient algorithm to learn the union of k-TSS languages. We evaluate our algorithm on an industrial dataset and thus demonstrate the relevance of our approach.

Keywords: grammatical inference · k-testable languages · union of languages · Galois connection

1 Introduction

A common problem in grammatical inference is to find, i.e. *learn*, a regular language from a set of examples of that language. When this set is divided into positive examples (belonging to the language) and negative examples (not belonging to the language), the problem is typically solved by searching for the smallest deterministic finite automaton (DFA) that accepts the positive examples, and rejects the negative ones. Moreover there exist algorithms which *identify in the limit* a DFA, that is, they eventually learn correctly any language/automaton from such examples [6].

We consider in this work a setting where one can observe positive examples from multiple different languages, but they are given together and it is not clear to which language each example belongs to. For example, given the following

* This research is supported by the Dutch Technology Foundation (STW) under the Robust CPS program (project 12693).

set of strings $S = \{aa, aaa, aaaa, abab, ababab,\ abba, abbba, abbbba\}$, learning a single automaton will be less informative than learning several DFAs encoding respectively the languages a^*, $(ab)^*$ and ab^*a. There is a trade-off between the number of languages and how specific each language should be. That is, covering all words through a single language may not be the desired result, but having a language for each word may also not be desired. The problem at hand is therefore double: to cluster the examples and learn the corresponding languages.

In this paper, we focus on k-testable languages in the strict sense (k-TSS) [10]. A k-TSS language is determined by a finite set of substrings of length at most k that are allowed to appear in the strings of the language. It has been proved that, unlike for regular languages, algorithms can learn k-TSS languages in the limit from text [16]. Practically, this learning guarantee has been used in a wide range of applications [2,3,12,13]. However, all these applications consider learning of a sole k-TSS language [2], or the training of several k-TSS languages in a context of supervised learning [13]. Learning unions of k-TSS languages has been suggested in [14].

A first contribution of this paper is a Galois connection between the lattice of all languages over alphabet Σ and the lattice of k-TSS languages over Σ. This result provides a unifying and abstract perspective on known properties of k-TSS languages, but also leads to several new insights. The Galois connection allows to give an alternative proof of the learnability in the limit of k-TSS languages, and suggests an algorithm for learning unions of k-TSS languages. A second contribution is the definition of a simple metric on k-TSS languages. Based on this metric, we define a clustering algorithm that allows us to efficiently learn unions of k-TSS languages.

Our research was initially motivated by a case study of print jobs that are submitted to large industrial printers. These print jobs can be represented by strings of symbols, where each symbol denotes a different media type, such as a book cover or a newspaper page. Together, this set of print jobs makes for a fairly complicated 'language'. Nevertheless, we observed that each print job can be classified as belonging to one of a fixed set of categories, such as 'book' or 'newspaper'. Two print jobs that belong to the same category are typically similar, to the extent that they only differ in terms of prefixes, infixes and suffixes. Therefore, the languages stand for the different families of print jobs. Our goal is to uncover these k-TSS languages.

This paper is organized as follows. In Section 2 we recall preliminary definitions on k-TSS languages and define a Galois connection that characterizes these languages. We then present in Section 3 our algorithm for learning unions of k-TSS languages. Finally, we report on the results we achieved for the industrial case study in Section 4. We refer to the full version of our paper for all the proofs.[1]

[1] For missing proofs, see `http://arxiv.org/abs/1812.08269`.

2 k-Testable Languages

The class of k-testable languages in the strict sense (k-TSS) has been introduced by McNaughton and Papert [10]. Informally, a k-TSS language is determined by a finite set of substrings of length at most k that are allowed to appear in the strings of the language. This makes it possible to use as a parser a sliding window of size k, which rejects the strings that at some point do not comply with the conditions. Concepts related to k-TSS languages have been widely used e.g. in information theory, pattern recognition and DNA sequence analysis [4,16]. Several definitions of k-TSS languages occur in the literature, but the differences are technical. In this section, we present a slight variation of the definition of k-TSS languages from [7], which in turn is a variation of the definition occurring in [4,5]. We establish a Galois connection that characterizes k-TSS languages, and show how this Galois connection may be used to infer a learning algorithm.

We write $\mathbb{N}$ to denote the set of natural numbers, and let i, j, k, m, and n range over $\mathbb{N}$.

2.1 Strings

Throughout this paper, we fix a finite set Σ of *symbols*. A *string* $x = a_1 \dots a_n$ is a finite sequence of symbols. The *length* of a string x, denoted $\mid x \mid$ is the number of symbols occurring in it. The empty string is denoted λ. We denote by Σ^* the set of all strings over Σ, and by Σ^+ the set of all nonempty strings over Σ (i.e. $\Sigma^* = \Sigma^+ \cup \{\lambda\}$). Similarly, we denote by $\Sigma^{<i}$, Σ^i and $\Sigma^{>i}$ the sets of strings over Σ of length less than i, equal to i, and greater than i, respectively.

Given two strings u and v, we will denote by $u \cdot v$ the concatenation of u and v. When the context allows it, $u \cdot v$ shall be simply written uv. We say that u is a *prefix* of v iff there exists a string w such that $uw = v$. Similarly, u is a *suffix* of v iff there exists a string w such that $wu = v$. We denote by $x[: k]$ the prefix of length k of x and $x[-k :]$ the suffix of length k of x.

A *language* is any set of strings, so therefore a subset of Σ^*. Concatenation is lifted to languages by defining $L \cdot L' = \{u \cdot v \mid u \in L \text{ and } v \in L'\}$. Again, we will write LL' instead of $L \cdot L'$ when the context allows it.

2.2 k-Testable Languages

A k-TSS language is determined by finite sets of strings of length $k-1$ or k that are allowed as prefixes, suffixes and substrings, respectively, together with all the short strings (with length at most $k-1$) contained in the language. The finite sets of allowed strings are listed in what McNaughton and Papert [10] called a *k-test vector*. The following definition is taken from [7], except that we have omitted the fixed alphabet Σ as an element in the tuple, and added a technical condition ($I \cap F = C \cap \Sigma^{k-1}$) that we need to prove Theorem 7.

Definition 1. *Let $k > 0$. A* k-test vector *is a 4-tuple $Z = \langle I, F, T, C \rangle$ where*

- *$I \subseteq \Sigma^{k-1}$ is a set of allowed* prefixes,

- $F \subseteq \Sigma^{k-1}$ *is a set of allowed* suffixes,
- $T \subseteq \Sigma^{k}$ *is a set of allowed* segments, *and*
- $C \subseteq \Sigma^{<k}$ *is a set of allowed* short strings *satisfying* $I \cap F = C \cap \Sigma^{k-1}$.

We write $\mathcal{T}_k$ *for the set of* k*-test vectors.*

Note that the set $\mathcal{T}_k$ of k-test vectors is finite. We equip set $\mathcal{T}_k$ with a partial order structure as follows.

Definition 2. *Let* $k > 0$. *The relation* $\sqsubseteq$ *on* $\mathcal{T}_k$ *is given by*

$$\langle I, F, T, C\rangle \sqsubseteq \langle I', F', T', C'\rangle \Leftrightarrow I \subseteq I' \text{ and } F \subseteq F' \text{ and } T \subseteq T' \text{ and } C \subseteq C'.$$

With respect to this ordering, $\mathcal{T}_k$ *has a least element* $\bot = \langle \emptyset, \emptyset, \emptyset, \emptyset\rangle$ *and a greatest element* $\top = \langle \Sigma^{k-1}, \Sigma^{k-1}, \Sigma^{k}, \Sigma^{<k}\rangle$. *The* union, intersection *and* symmetric difference *of two* k*-test vectors* $Z = \langle I, F, T, C\rangle$ *and* $Z' = \langle I', F', T', C'\rangle$ *are given by, respectively,*

$$\begin{aligned}
Z \sqcup Z' &= \langle I \cup I', F \cup F', T \cup T', C \cup C' \cup (I \cap F') \cup (I' \cap F)\rangle,\\
Z \sqcap Z' &= \langle I \cap I', F \cap F', T \cap T', C \cap C'\rangle,\\
Z \mathbin{\triangle} Z' &= \langle I \mathbin{\triangle} I', F \mathbin{\triangle} F', T \mathbin{\triangle} T', C \mathbin{\triangle} C' \mathbin{\triangle} (I' \cap F) \mathbin{\triangle} (I \cap F')\rangle.
\end{aligned}$$

The reader may check that $Z \sqcup Z'$, $Z \sqcap Z'$ and $Z \mathbin{\triangle} Z'$ are k-test vectors indeed, preserving the property $I \cap F = C \cap \Sigma^{k-1}$. The reader may also check that $(\mathcal{T}_k, \sqsubseteq)$ is a lattice with $Z \sqcup Z'$ the least upper bound of Z and Z', and $Z \sqcap Z'$ the greatest lower bound of Z and Z'. The symmetric difference operation $\mathbin{\triangle}$ will be used further on to define a metric on k-test vectors.

We can associate a k-test vector $\alpha_k(L)$ to each language L by taking all prefixes of length $k-1$ of the strings in L, all suffixes of length $k-1$ of the strings in L , and all substrings of length k of the strings in L. Any string which is both an allowed prefix and an allowed suffix is also a short string, as well as any string in L with length less than $k-1$.

Definition 3. *Let* $L \subseteq \Sigma^*$ *be a language and* $k \in \mathbb{N}$. *Then* $\alpha_k(L)$ *is the* k*-test vector* $\langle I_k(L), F_k(L), T_k(L), C_k(L)\rangle$ *where*

- $I_k(L) = \{u \in \Sigma^{k-1} \mid \exists v \in \Sigma^* : uv \in L\}$,
- $F_k(L) = \{w \in \Sigma^{k-1} \mid \exists v \in \Sigma^* : vw \in L\}$,
- $T_k(L) = \{v \in \Sigma^{k} \mid \exists u, w \in \Sigma^* : uvw \in L\}$, *and*
- $C_k(L) = (L \cap \Sigma^{<k-1}) \cup (I_k(L) \cap F_k(L))$.

It is easy to see that operation $\alpha_k : 2^{\Sigma^*} \to \mathcal{T}_k$ is monotone.

Proposition 4. *For all languages* L, L' *and for all* $k > 0$,

$$L \subseteq L' \Rightarrow \alpha_k(L) \sqsubseteq \alpha_k(L').$$

Conversely, we associate a language $\gamma_k(Z)$ to each k-test vector $Z = \langle I, F, T, C\rangle$, consisting of all the short strings from C together with all strings of length at least $k-1$ whose prefix of length $k-1$ is in I, whose suffix of length $k-1$ is in F, and where all substrings of length k belong to T.

Definition 5. *Let $Z = \langle I, F, T, C\rangle$ be a k-test vector, for some $k > 0$. Then*

$$\gamma_k(Z) = C \cup ((I\Sigma^* \cap \Sigma^* F) \setminus (\Sigma^*(\Sigma^k \setminus T)\Sigma^*)).$$

*We say that a language L is k-*testable in the strict sense *(k-TSS) if there exists a k-test vector Z such that $L = \gamma_k(Z)$. Note that all k-TSS languages are regular.*

Again, it is easy to see that operation $\gamma_k : \mathcal{T}_k \to 2^{\Sigma^*}$ is monotone.

Proposition 6. *For all $k > 0$ and for all k-test vectors Z and Z',*

$$Z \sqsubseteq Z' \Rightarrow \gamma_k(Z) \subseteq \gamma_k(Z').$$

The next theorem, which is our main result about k-testable languages, asserts that α_k and γ_k form a (monotone) Galois connection [11] between lattices $(\mathcal{T}_k, \sqsubseteq)$ and $(2^{\Sigma^*}, \subseteq)$.

Theorem 7 (Galois connection). *Let $k > 0$, let $L \subseteq \Sigma^*$ be a language, and let Z be a k-test vector. Then $\alpha_k(L) \sqsubseteq Z \;\Leftrightarrow\; L \subseteq \gamma_k(Z)$.*

The above theorem generalizes results on strictly k-testable languages from [4,16]. Composition $\gamma_k \circ \alpha_k$ is commonly called the *associated closure operator*, and composition $\alpha_k \circ \gamma_k$ is known as the *associated kernel operator*. The fact that we have a Galois connection has some well-known consequences for these associated operators.

Corollary 8. *For all $k > 0$, $\gamma_k \circ \alpha_k$ and $\alpha_k \circ \gamma_k$ are monotone and idempotent.*

Monotony of $\gamma_k \circ \alpha_k$ was established previously as Theorem 3.2 in [4] and as Lemma 3.3 in [16].

Corollary 9. *For all $k > 0$, $L \subseteq \Sigma^*$ and $Z \in \mathcal{T}_k$,*

$$\alpha_k \circ \gamma_k(Z) \sqsubseteq Z \tag{1}$$

$$L \subseteq \gamma_k \circ \alpha_k(L) \tag{2}$$

Inequality (1) asserts that the associated kernel operator $\alpha_k \circ \gamma_k$ is *deflationary*, while inequality (2) says that the associated closure operator $\gamma_k \circ \alpha_k$ is *inflationary* (or *extensive*). Inequality (2) was established previously as Lemma 3.1 in [4] and (also) as Lemma 3.1 in [16].

Another immediate corollary of the Galois connection is that in fact $\gamma_k \circ \alpha_k(L)$ is the smallest k-TSS language that contains L. This has been established previously as Theorem 3.1 in [4].

Corollary 10. *For all $k > 0$, $L \subseteq \Sigma^*$, and $Z \in \mathcal{T}_k$,*

$$L \subseteq \gamma_k(Z) \Rightarrow \gamma_k \circ \alpha_k(L) \subseteq \gamma_k(Z).$$

As a final corollary, we mention that $\alpha_k \circ \gamma_k(Z)$ is the smallest k-test vector that denotes the same language as Z. This is essentially Lemma 1 of [16].

Corollary 11. *For all $k > 0$ and $Z \in \mathcal{T}_k$, $\gamma_k \circ \alpha_k \circ \gamma_k(Z) = \gamma_k(Z)$. Moreover, for any $Z' \in \mathcal{T}_k$,*

$$\gamma_k(Z) = \gamma_k(Z') \Rightarrow \alpha_k \circ \gamma_k(Z) \sqsubseteq Z'.$$

We can provide a simple characterization of $\alpha_k \circ \gamma_k(Z)$ as the *canonical* k-test vector obtained by removing all the allowed prefixes, suffixes and segments that do not occur in the k-testable language generated by Z.

Definition 12. *Let $Z = \langle I, F, T, C\rangle$ be a k-test vector, for some $k > 0$. We say that $u \in I$ is a* junk prefix *of Z if u does not occur as a prefix of any string in $\gamma_k(Z)$. Similarly, we say that $u \in F$ is a* junk suffix *of Z if u does not occur as a suffix of any string in $\gamma_k(Z)$, and we say that $u \in T$ is a* junk segment *of Z if u does not occur as a substring of any string in $\gamma_k(Z)$. We call Z* canonical *if it does not contain any junk prefixes, junk suffixes, or junk segments.*

Proposition 13. *Let Z be a k-test vector, for some $k > 0$, and let Z' be the canonical k-test vector obtained from Z by deleting all junk prefixes, junk suffixes, and junk segments. Then $\alpha_k \circ \gamma_k(Z) = Z'$.*

Proposition 13 implies that if we restrict the lattice $(\mathcal{T}_k, \sqsubseteq)$ to the canonical k-test vectors, our Galois connection becomes a Galois insertion.

2.3 Learning k-TSS Languages

It is well-known that any k-TSS language can be identified in the limit from positive examples [4,5]. Below we recall the basic argument; we refer to [4,5,16] for efficient algorithms.

Theorem 14. *Any k-TSS language can be identified in the limit from positive examples.*

Proof. Let L be a k-TSS language and let $w_1, w_2, w_3, \ldots$ be an enumeration of L. Let $L_0 = \emptyset$ and $L_i = L_{i-1} \cup \{w_i\}$, for $i > 0$. We then have

$$L_1 \subseteq L_2 \subseteq L_3 \subseteq \cdots$$

By monotonicity of α_k (Proposition 4) we obtain

$$\alpha_k(L_1) \sqsubseteq \alpha_k(L_2) \sqsubseteq \alpha_k(L_3) \sqsubseteq \cdots \tag{3}$$

and by monotonicity of γ_k (Proposition 6)

$$\gamma_k \circ \alpha_k(L_1) \subseteq \gamma_k \circ \alpha_k(L_2) \subseteq \gamma_k \circ \alpha_k(L_3) \subseteq \cdots \tag{4}$$

Since $\gamma_k \circ \alpha_k$ is inflationary (Corollary 9), L is a k-TSS language and, for each i, $\gamma_k \circ \alpha_k(L_i)$ is the smallest k-TSS language that contains L_i (Corollary 10), we have

$$L_i \subseteq \gamma_k \circ \alpha_k(L_i) \subseteq L \tag{5}$$

Because $(\mathcal{T}_k, \sqsubseteq)$ is a finite partial order it does not have an infinite ascending chain. This means that sequence (3) converges. But then sequence (4) also converges, that is, there exists an n such that, for all $m \geq n$, $\gamma_k \circ \alpha_k(L_m) = \gamma_k \circ \alpha_k(L_n)$. By equations (4) and (5) we obtain, for all i,

$$L_i \subseteq \gamma_k \circ \alpha_k(L_i) \subseteq \gamma_k \circ \alpha_k(L_n) \subseteq L$$

This implies $L = \gamma_k \circ \alpha_k(L_n)$, meaning that the sequence (4) of k-TSS languages converges to L.

3 Learning Unions of k-TSS Languages

In this section, we present guarantees concerning learnability in the limit of unions of k-TSS languages. Then, we present an algorithm merging closest and compatible k-TSS languages.

3.1 Generalities

It is well-known that the class of k-testable languages in the strict sense is not closed under union. Take for instance the two 3-testable languages, represented by their DFA's in Figure 1a, that are generated by the following 3-test vectors:

$$Z = \langle \{aa\}, \{aa\}, \{aaa\}, \{aa\} \rangle$$
$$Z' = \langle \{ba, bb\}, \{ab, bb\}, \{baa, bab, aaa, aab\}, \{bb\} \rangle$$

with $\Sigma = \{a, b\}$. The union $\gamma_3(Z) \cup \gamma_3(Z')$ of these languages, represented by its DFA in Figure 1b, is not a 3-testable language. Indeed, it is not a k-testable language for any value of $k > 0$. For $k = 1$, the only k-testable language that extends $\gamma_3(Z) \cup \gamma_3(Z')$ is Σ^*. For $k \geq 2$, the problem is that since a^{k-1} is an allowed prefix, $a^{k-1}b$ is an allowed segment, and $a^{k-2}b$ is an allowed suffix, $a^{k-1}b$ has to be in the language, even though it is not an element of $\gamma_3(Z) \cup \gamma_3(Z')$.

It turns out that we can generalize Theorem 14 to unions of k-TSS languages.

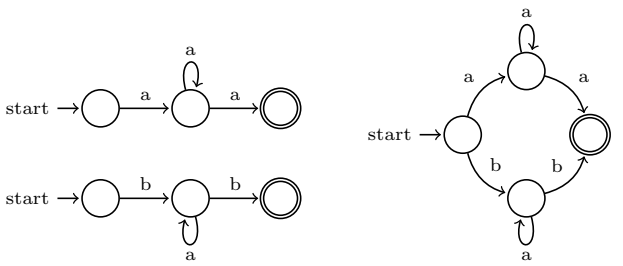
Theorem 15. *Any language that is a union of k-TSS languages can be identified in the limit from positive examples.*

Proof. Let $L = L_1 \cup \cdots \cup L_l$, where all the L_p are k-TSS languages, and let $w_1, w_2, w_3, \ldots$ be an enumeration of L. Define, for $i > 0$,

$$K_i = \bigcup_{j=1}^{i} \gamma_k \circ \alpha_k(\{w_j\}).$$

Since each w_j is included in a k-TSS language contained in L, and $\gamma_k \circ \alpha_k(\{w_j\})$ is the smallest k-TSS language that contains w_j, we conclude that, for all j, $\gamma_k \circ \alpha_k(\{w_j\}) \subseteq L$, which in turn implies $K_i \subseteq L$. Since there are only finitely many k-test vectors and finitely many k-TSS languages, the sequence

$$K_1 \subseteq K_2 \subseteq K_3 \subseteq \cdots \tag{6}$$



(a) $\gamma_3(Z)$ and $\gamma_3(Z')$. (b) $\gamma_3(Z) \cup \gamma_3(Z')$.

Fig. 1. k-testable languages are not closed under union.

converges, that is there exists an n such that, for all $m \geq n$, $K_m = K_n$. This implies that all w_j are included in K_n, that is $L \subseteq K_n$. In combination with the above observation that all K_i are contained in L, this proves that sequence (6) converges to L.

The proof of Theorem 15 provides us with a simple first algorithm to learn unions of k-TSS languages: for each example word that we see, we compute the k-test vector and then we take the union of the languages denoted by all those k-test vectors. The problem with this algorithm is that potentially we end up with a huge number of different k-test vectors. Thus we would like to cluster as many k-test vectors in the union as we can, without changing the overall language. Before we can introduce our clustering algorithm, we first need to define a metric on k-test vectors.

Definition 16. *The* cardinality *of a k-test vector $Z = \langle I, F, T, C \rangle$ is defined by:*

$$|Z| = |I| + |F| + |T| + |C \cap \Sigma^{<k-1}|.$$

Intuitively, the distance between two k-test vectors is the number of prefixes, suffixes, substrings and short words that must be added/removed to transform one k-test vector into the other. For examples, see Fig. 2b.

Definition 17. *The function $d \colon \mathcal{T}_k \times \mathcal{T}_k \mapsto \mathbb{R}^+$, which defines the* distance *between a pair of k-test vectors, is given by: $d(Z, Z') = |Z \mathbin{\triangle} Z'|$.*

The next proposition provides a necessary and sufficient condition for when the γ_k operator preserves least upper bounds, that is, when the union of the languages of two k-test vectors equals the language of the union of these vectors. The basic idea is that, for each k-test vector, we may construct a directed graph in which the segments are the nodes. The graph contains an edge from segment u to segment v if, when the content of the sliding window is u at some point, it may become v when the sliding window advances one step. There exists a 1-to-1 correspondence between paths in this graph from an initial segment to a final segment, and strings in the associated language with length at least $k - 1$. Given

two test vectors Z and Z', we consider the graph for the union $Z \sqcup Z'$. The union of the languages of Z and Z' equals the language of $Z \sqcup Z'$ iff in this graph there exists no path from a node in $Z \setminus Z'$ to a node in $Z' \setminus Z$, or vice versa. Such a path would allow us to construct a word in the language of $Z \sqcup Z'$ that is neither in the language of Z nor in the language of Z'.

Proposition 18. *Suppose $Z = \langle I, F, T, C\rangle$ and $Z' = \langle I', F', T', C'\rangle$ are canonical k-test vectors, for some k. Let $\bullet \notin \Sigma$ be a fresh symbol, and let $G = (V, E)$ be the directed graph with*

$$V = \{\bullet u \mid u \in I \cup I'\} \cup T \cup T' \cup \{u\bullet \mid u \in F \cup F'\},$$
$$E = \{(au, ub) \in V \times V \mid a, b \in \Sigma \cup \{\bullet\}, u \in \Sigma^{k-1}\}.$$

Suppose each vertex in V is colored either red, blue or white. Vertices in $T \setminus T'$ are red, vertices in $T' \setminus T$ are blue, and vertices in $T \cap T'$ are white. A vertex $\bullet u$ is red if $u \in I \setminus I'$, blue if $u \in I' \setminus I$, and white if $u \in I \cap I'$. A vertex $u\bullet$ is red if $u \in F \setminus F'$, blue if $u \in F' \setminus F$, and white if $u \in F \cap F'$. Then $\gamma_k(Z \sqcup Z') = \gamma_k(Z) \cup \gamma_k(Z')$ iff there exists no path in G from a red vertex to a blue vertex, nor from a blue vertex to a red vertex.

Suppose alphabet Σ contains n elements. Then the size of graph G from Proposition 18 is in $O(n \cdot |Z \cup Z'|)$, and we can construct G from Z and Z' in time $O((n+k) \cdot |Z \cup Z'|)$. Since the reachability property in Proposition 18 can be decided in a time that is linear in the size of G, we obtain an $O((n+k) \cdot |Z \cup Z'|)$-time algorithm for deciding $\gamma_k(Z \sqcup Z') = \gamma_k(Z) \cup \gamma_k(Z')$.

3.2 Efficient algorithm

Our algorithm to learn unions of k-testable languages is based on hierarchical clustering. Given a set $\mathcal{S}$ of n words, we compute its related set of k-test vectors $S = \{\alpha_k(\{x\}) \mid x \in \mathcal{S}\}$. Note that the k-test vectors are canonical. Then, an $n \times n$ distance matrix is computed. To that end, the distance used is the pairwise distance between k-test vectors defined in Definition 17. Next, the algorithm finds the closest pair of compatible k-test vectors Z and Z', such that $\gamma_k(Z \sqcup Z') = \gamma_k(Z) \cup \gamma_k(Z')$ and computes their union. An efficient implementation for finding closest k-test vectors is the nearest-neighbor chain algorithm [1], which finds pairs of k-test vectors such that these two closest k-test vectors are the nearest neighbors of each other. The distance between the merged k-test vectors and the remaining k-test vectors in S is updated. These two operations are repeated until all initial k-test vectors have been merged into one, or that no allowed union of two k-test vectors such that $\gamma_k(Z \sqcup Z') = \gamma_k(Z) \cup \gamma_k(Z')$ is possible. We gather at the end of the process a linkage between k-test vectors, which can lead to the computation of a dendrogram. When the number of k-test vectors to learn is known, one can use this expected number of languages to find the threshold that would, given the hierarchical clustering, return the desired unions of k-test vectors.

Example 19. Let $k = 3$. Given the sample of strings $\mathcal{S}$ in Table 2a, compute the associate sample of 3-test vectors $S = \{Z_1, Z_2, \ldots, Z_8\}$. Then, compute its distance matrix (Table 2b) using the metric defined in Definition 17. Using classical linkage algorithms (for instance nearest-neighbor chain algorithm), compute the related linkage matrix depicted in Table 2c. We gather the dendrogram shown in Figure 2d, where the 3 remaining 3-test vectors $Z_1 \sqcup Z_8$, $Z_2 \sqcup Z_5 \sqcup Z_7$ and $Z_3 \sqcup Z_4 \sqcup Z_6$ cannot be merged. Indeed:

- $\gamma_k(Z_1 \sqcup Z_8 \sqcup Z_2 \sqcup Z_5 \sqcup Z_7) \neq \gamma_k(Z_1 \sqcup Z_8) \cup \gamma_k(Z_2 \sqcup Z_5 \sqcup Z_7)$.
- $\gamma_k(Z_1 \sqcup Z_8 \sqcup Z_3 \sqcup Z_4 \sqcup Z_6) \neq \gamma_k(Z_1 \sqcup Z_8) \cup \gamma_k(Z_3 \sqcup Z_4 \sqcup Z_6)$.
- $\gamma_k(Z_2 \sqcup Z_5 \sqcup Z_7 \sqcup Z_3 \sqcup Z_4 \sqcup Z_6) \neq \gamma_k(Z_2 \sqcup Z_5 \sqcup Z_7) \cup \gamma_k(Z_3 \sqcup Z_4 \sqcup Z_6)$.

With a desired number of 3-TSS languages to learn of 3, the returned languages are $\gamma_k(Z_1 \sqcup Z_8)$ and $\gamma_k(Z_2 \sqcup Z_5 \sqcup Z_7)$ and $\gamma_k(Z_3 \sqcup Z_4 \sqcup Z_6)$. With a desired number of 3-TSS languages to learn of 4, the returned languages would be $\gamma_k(Z_1 \sqcup Z_8)$ and $\gamma_k(Z_2 \sqcup Z_5 \sqcup Z_7)$ and $\gamma_k(Z_3)$ and $\gamma_k(Z_4 \sqcup Z_6)$ instead.

$\mathcal{S}$	S
baba	$Z_1 = \langle\{ba\}, \{ba\}, \{bab, aba\}, \{\}\rangle$
abba	$Z_2 = \langle\{ab\}, \{ba\}, \{abb, bba\}, \{\}\rangle$
abcabc	$Z_3 = \langle\{ab\}, \{bc\}, \{abc, bca, cab\}, \{\}\rangle$
cbacba	$Z_4 = \langle\{cb\}, \{ba\}, \{cba, bac, acb\}, \{\}\rangle$
abbbba	$Z_5 = \langle\{ab\}, \{ab\}, \{abb, bbb, bba\}, \{\}\rangle$
cbacbacba	$Z_6 = \langle\{cb\}, \{ba\}, \{cba, bac, acb\}, \{\}\rangle$
abbba	$Z_7 = \langle\{ab\}, \{ba\}, \{abb, bbb, bba\}, \{\}\rangle$
bababbc	$Z_8 = \langle\{ba\}, \{bc\}, \{bab, aba, abc\}, \{\}\rangle$

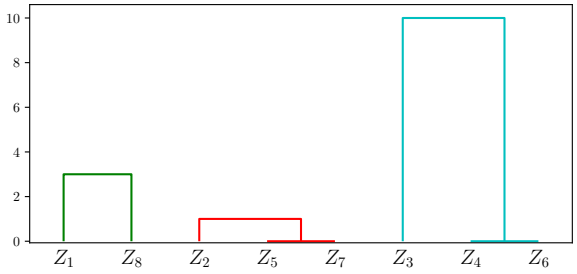
(a) Dataset and corresponding 3-test vectors.

	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8
Z_1	0	6	9	7	7	7	7	3
Z_2	6	0	7	7	1	7	1	9
Z_3	9	7	0	10	8	10	8	6
Z_4	7	7	10	0	8	0	8	10
Z_5	7	1	8	8	0	8	0	10
Z_6	7	7	10	0	8	0	8	10
Z_7	7	1	8	8	0	8	0	10
Z_8	3	9	6	10	10	10	10	0

(b) Distance matrix.

Z_5	Z_7	0
Z_4	Z_6	0
Z_2	$Z_5 \sqcup Z_7$	1
Z_1	Z_8	3
Z_3	$Z_4 \sqcup Z_6$	10

(c) Linkage matrix.



(d) Corresponding dendrogram.

Fig. 2. Learning unions of k-test vectors.

We can see here that the lower bound on the number of returned languages is the number of unions of k-test vectors satisfying the compatibility constraint

$\gamma_k(Z \sqcup Z') = \gamma_k(Z) \cup \gamma_k(Z')$. However, in case this constraint is relaxed, it is possible to obtain a clustering into less parts, up to a single cluster standing for $\gamma_k(\bigsqcup_{Z \in S})$.

4 Case Study

Job dataset Our case study has been inspired by an industrial problem related to the domain of cyber-physical systems. Recent work [15] focused on the impact of design parameters of a flexible manufacturing system on its productivity. It appeared in the aforementioned study that the productivity depends on the jobs being rendered. To that end, the prior identification of the different job patterns is crucial to enabling engineers to optimize parameters related to the flexible manufacturing system.

Table 1. Sample of identified job patterns.

job	pattern	3-test vector	type of job
$aaaaa$	a^+	$Z = \langle\{aa\}, \{aa\}, \{aaa\}, \{aa\}\rangle$	homogeneous
$aaaaaaaaaa$			
$aaaaa \ldots aaa$			
$abababab$	$(ab)^+$	$Z = \langle\{ab\}, \{ab\}, \{aba, bab\}, \{ab\}\rangle$	heterogeneous
$abababababab$			
$abcabcabc$	$(abc)^+$	$Z = \langle\{ab\}, \{bc\}, \{abc, bca, cab\}, \emptyset\rangle$	
$abcabcabcabcabc$			
$abcbcbcbca$	$a(bc)^+a$	$Z = \langle\{ab\}, \{ca\}, \{abc, bcb, cbc, cba\}, \emptyset\rangle$	miscellaneous

We consider a dataset containing strings, each representing a job. Our job patterns are also represented by 3-testable languages, the 3-test vectors of which are shown in Table 1. Our dataset, implementations and complete results are available[2].

5 Conclusion

In this paper, we defined a Galois connection characterizing k-testable languages. We also described an efficient algorithm to learn unions of k-testable languages that results from this Galois connection. From a practical perspective, we see that obtaining more than one representation is meaningful since a too generalized solution is not necessarily the best. To avoid unnecessary generalizations, the union of two k-testable languages that would not be a k-testable language is

[2] See `https://gitlab.science.ru.nl/alinard/learning-union-ktss`

not allowed. Note also that depending on the applications, expert knowledge can provide an indication on the number of languages the returned union should contain. In further work, we would like to extend the learning of unions of languages to regular languages. An attempt to learn pairwise disjoint regular languages has been made in [8,9]. However, no learnability guarantee has been provided so far.

References

1. Benzécri, J.P.: Construction d'une classification ascendante hiérarchique par la recherche en chaîne des voisins réciproques. Les cahiers de l'analyse des données **7**(2), 209–218 (1982)
2. Bex, G.J., Neven, F., Schwentick, T., Tuyls, K.: Inference of concise dtds from xml data. In: Proceedings of the 32nd international conference on Very large data bases. pp. 115–126 (2006)
3. Coste, F.: Learning the language of biological sequences. In: Topics in Grammatical Inference, pp. 215–247. Springer (2016)
4. García, P., Vidal, E.: Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. IEEE Transactions on Pattern Analysis & Machine Intelligence **12**(9), 920–925 (1990)
5. Garcia, P., Vidal, E., Oncina, J.: Learning locally testable languages in the strict sense. In: Algorithmic Learning Theory (ALT), First International Workshop. pp. 325–338 (1990)
6. Gold, M.: Language identification in the limit. Information Control **10**(5), 447–474 (1967)
7. de la Higuera, C.: Grammatical inference: learning automata and grammars. Cambridge University Press (2010)
8. Linard, A.: Learning several languages from labeled strings: State merging and evolutionary approaches. arXiv preprint arXiv:1806.01630 (2018)
9. Linard, A., Smetsers, R., Vaandrager, F., Waqas, U., van Pinxten, J., Verwer, S.: Learning pairwise disjoint simple languages from positive examples. arXiv preprint arXiv:1706.01663 (2017)
10. McNaughton, R., Papert, S.A.: Counter-Free Automata (M.I.T. Research Monograph No. 65). The MIT Press (1971)
11. Nielson, F., Nielson, H., Hankin, C.: Principles of Program Analysis. Springer-Verlag, Berlin Heidelberg (1999)
12. Rogers, J., Pullum, G.K.: Aural pattern recognition experiments and the subregular hierarchy. Journal of Logic, Language and Information **20**(3), 329–342 (2011)
13. Tantini, F., Terlutte, A., Torre, F.: Sequences classification by least general generalisations. In: International Colloquium on Grammatical Inference. pp. 189–202. Springer (2010)
14. Torres, I., Varona, A.: k-tss language models in speech recognition systems. Computer Speech & Language **15**(2), 127–148 (2001)
15. Umar, W., Geilen, M., Stuijk, S., van Pinxten, J., Basten, T., Somers, L., Corporaal, H.: A fast estimator of performance with respect to the design parameters of self reentrant flowshops. In: Euromicro Conference on Digital System Design. pp. 215–221 (2016)
16. Yokomori, T., Kobayashi, S.: Learning local languages and their application to dna sequence analysis. IEEE Transactions on Pattern Analysis & Machine Intelligence **20**(10), 1067–1079 (1998)