Separability by Piecewise Testable Languages is PTime-Complete

Tomáš Masopust

Institute of Mathematics, Czech Academy of Sciences, Žižkova 22, 616 62 Brno, Czechia

Abstract

Piecewise testable languages form the first level of the Straubing-Thérien hierarchy. The membership problem for this level is decidable and testing if the language of a DFA is piecewise testable is NL-complete. The question has not yet been addressed for NFAs. We fill in this gap by showing that it is PSpace-complete. The main result is then the lower-bound complexity of separability of regular languages by piecewise testable languages. Two regular languages are separable by a piecewise testable language if the piecewise testable language includes one of them and is disjoint from the other. For languages represented by NFAs, separability by piecewise testable languages is known to be decidable in PTime. We show that it is PTime-hard and that it remains PTime-hard even for minimal DFAs.

Keywords: Separability, piecewise testable languages, complexity
2010 MSC: 68Q45, 68Q17, 68Q25, 03D05

1. Introduction

A regular language over Σ is *piecewise testable* if it is a finite boolean combination of languages of the form $\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \cdots \Sigma^* a_n \Sigma^*$, where $a_i \in \Sigma$ and $n \geq 0$. If n is bounded by a constant k, then the language is called *k-piecewise testable*. Piecewise testable languages are exactly those regular languages whose syntactic monoid is $\mathcal{J}$-trivial [36]. Simon [37] provided various characterizations of piecewise testable languages, e.g., in terms of monoids or automata. These languages are of interest in many disciplines of mathematics, such as semigroup theory [2, 3, 28] for their relation to Green's relations or in logic on words [10] for their relation to first-order logic FO[<] and the *Straubing-Thérien hierarchy* [40, 43].

For an alphabet Σ, level 0 of the Straubing-Thérien hierarchy is defined as $\mathscr{L}(0) = \{\emptyset, \Sigma^*\}$. For integers $n \geq 0$, level $\mathscr{L}(n + \frac{1}{2})$ consists of all finite unions of languages $L_0 a_1 L_1 a_2 \ldots a_k L_k$ with $k \geq 0$, $L_0, \ldots, L_k \in \mathscr{L}(n)$, and $a_1, \ldots, a_k \in \Sigma$, and level $\mathscr{L}(n+1)$ consists of all finite Boolean combinations of languages from level $\mathscr{L}(n + \frac{1}{2})$. The levels of the hierarchy contain only *star-free* languages [27]. Piecewise testable languages form the first level of the hierarchy. The hierarchy does not collapse on any level [5]. In spite of a recent development [1, 29, 32], deciding whether a language belongs to level ℓ of the hierarchy is open for $\ell > \frac{7}{2}$. The Straubing-Thérien hierarchy is further closely related to the *dot-depth hierarchy* [5, 7, 23, 41] and to complexity theory [45].

The fundamental question is how to efficiently recognize whether a given regular language is piecewise testable. Stern [39] provided a solution that was later improved by Trahtman [44] and Klíma and Polák [21]. Stern presented an algorithm deciding piecewise testability of a regular language represented by a DFA in time $O(n^5)$, where n is the number of states of the DFA. Trahtman improved Stern's algorithm to time quadratic with respect to the number of states and linear with respect to the size of the alphabet, and Klíma and Polák found an algorithm for DFAs that is quadratic with respect to the size of the alphabet and linear with respect to the number of states. Cho and Huynh [6] proved that deciding piecewise testability for DFAs is NL-complete. Event though the complexity for DFAs has intensively been investigated, a study

Email address: `masopust@math.cas.cz` (Tomáš Masopust)

for NFAs is missing in the literature. We fill in this gap by showing that deciding piecewise testability for NFAs is PSpace-complete (Theorem 2).

The knowledge of the minimal k or a reasonable bound on k for which a piecewise testable language is k-piecewise testable is of interest in several applications [24, 16]. The complexity of finding the minimal k has been investigated in the literature [16, 20, 21, 26]. Testing whether a piecewise testable language is k-piecewise testable is coNP-complete for $k \geq 4$ if the language is represented as a DFA [20] and PSpace-complete if the language is represented as an NFA [26]. The complexity for DFAs and $k < 4$ has also been discussed in detail [26]. Klíma and Polák [21] further showed that the upper bound on k is given by the depth of the minimal DFA. This result has recently been generalized to NFAs [25].

The recent interest in piecewise testable languages is mainly because of the applications of separability of regular languages by piecewise testable languages in logic on words [31] and in XML schema languages [8, 16, 24]. Given two languages K and L and a family of languages $\mathcal{F}$, the *separability problem* asks whether there exists a language S in $\mathcal{F}$ such that S includes one of the languages K and L and is disjoint from the other. Place and Zeitoun [31] used separability to obtain new decidability results of the membership problem for some levels of the Straubing-Thérien hierarchy. The separability problem for regular languages represented by NFAs and the family of piecewise testable languages is decidable in polynomial time with respect to both the number of states and the size of the alphabetby [8, 30]. Separability by piecewise testable languages is of interest also outside regular languages. Although separability of context-free languages by regular languages is undecidable [17], separability by piecewise testable languages is decidable (even for some non-context-free languages) [9]. Piecewise testable languages are further investigated in natural language processing [11, 33], cognitive and sub-regular complexity [34], and learning theory [12, 22]. They have been extended from word languages to tree languages [4, 13, 14].

In this paper, we show that separability of regular languages represented as NFAs by piecewise testable languages is a PTime-complete problem (Theorem 3) and that it remains PTime-hard even for minimal DFAs. Consequently, the separability problem is unlikely to be solvable in logarithmic space or effectively parallelizable.

2. Preliminaries

We assume that the reader is familiar with automata theory [38]. The cardinality of a set A is denoted by $|A|$ and the power set of A by 2^A. The free monoid generated by an alphabet Σ is denoted by Σ^*. A word over Σ is any element of Σ^*; the empty word is denoted by ε. For a word $w \in \Sigma^*$, $\mathrm{alph}(w) \subseteq \Sigma$ denotes the set of all symbols occurring in w.

A *nondeterministic finite automaton* (NFA) is a quintuple $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$, where Q is the finite nonempty set of states, Σ is the input alphabet, $Q_0 \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of accepting states, and $\delta\colon Q \times \Sigma \to 2^Q$ is the transition function extended to the domain $2^Q \times \Sigma^*$ in the usual way. The language *accepted* by $\mathcal{A}$ is the set $L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(Q_0, w) \cap F \neq \emptyset\}$.

A *path* π from a state q_0 to a state q_n under a word $a_1a_2\cdots a_n$, for some $n \geq 0$, is a sequence of states and input symbols $q_0, a_1, q_1, a_2, \ldots, q_{n-1}, a_n, q_n$ such that $q_{i+1} \in \delta(q_i, a_{i+1})$, for all $i = 0, 1, \ldots, n-1$. Path π is *accepting* if $q_0 \in Q_0$ and $q_n \in F$. We write $q_0 \xrightarrow{a_1a_2\cdots a_n} q_n$ to denote that there is a path from q_0 to q_n under the word $a_1a_2\cdots a_n$.

We says that $\mathcal{A}$ has a *cycle over an alphabet* $\Gamma \subseteq \Sigma$ if there is a state q in $\mathcal{A}$ and a word w over Σ such that $q \xrightarrow{w} q$ and $\mathrm{alph}(w) = \Gamma$.

The NFA $\mathcal{A}$ is *deterministic* (DFA) if $|Q_0| = 1$ and $|\delta(q, a)| = 1$ for every $q \in Q$ and $a \in \Sigma$. Although we define DFAs as complete, we mostly depict only the most important transitions in our illustrations. The reader can easily complete such an incomplete DFA.

Let K and L be languages. A language S *separates* K *from* L if S contains K and does not intersect L. Languages K and L are *separable by a family of languages* $\mathcal{F}$ if there exists a language S in $\mathcal{F}$ that separates K from L or L from K.

3. Piecewise Testability for NFAs

Given an NFA $\mathcal{A}$ over an alphabet Σ, the *piecewise-testability problem* asks whether the language $L(\mathcal{A})$ is piecewise testable. Although the membership in PSPACE follows basically from the result by Cho and Huynh [6], we prefer to provide the proof here for two reasons: (i) we would like to provide unfamiliar readers with a method to recognize whether a regular language is piecewise testable, (ii) Cho and Huynh assume that the input is a minimal DFA, hence it is necessary to extend their algorithm with a non-equivalence check. We use the following characterization in our proof.

Proposition 1 (Simon [37], Cho and Huynh [6, Proposition 2.3(b)]). *A regular language L is not piecewise testable if and only if the minimal DFA for L either*

1. *contains a nontrivial (non-self-loop) cycle or*
2. *there are three distinct states p, q, q' such that q and q' are reachable from p by words over the symbols that form self-loops on both q and q'; formally, there are paths $p \xrightarrow{w} q$ and $p \xrightarrow{w'} q'$ in the DFA with $w, w' \in \Sigma(q) \cap \Sigma(q')$, where $\Sigma(q) = \{a \in \Sigma \mid q \xrightarrow{a} q\}$.*

We now prove the first result of this paper.

Theorem 2. *The piecewise-testability problem for NFAs is* PSPACE*-complete.*

Proof. To prove that piecewise testability is in PSPACE, let $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ be an NFA. Since $\mathcal{A}$ is nondeterministic, we cannot directly use the algorithm of Cho and Huynh [6]. Instead, we consider the DFA $\mathcal{A}'$ obtained from $\mathcal{A}$ by the standard subset construction where the states of $\mathcal{A}'$ are subsets of states of $\mathcal{A}$. We now need to modify Cho and Huynh's algorithm to check whether the guessed states are distinguishable. For a set of states $X \subseteq Q$, let $\Sigma(X) = \{a \in \Sigma \mid X \xrightarrow{a} X\}$. The entire algorithm is presented as Algorithm 1.

Algorithm 1: Non-piecewise testability (symbol $\rightsquigarrow$ stands for reachability)

Input : An NFA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$
Output: `true` if and only if $L(\mathcal{A})$ is not piecewise testable

1 Guess states $X, Y \subseteq Q$ of $\mathcal{A}'$ `// Verify property (1)`
2 **if** $Q_0 \rightsquigarrow X \rightsquigarrow Y \rightsquigarrow X$ **then** go to line 10
3 Guess states $P, X, Y \subseteq Q$ of $\mathcal{A}'$ `// Verify property (2)`
4 Check that $Q_0 \rightsquigarrow P$, $Q_0 \rightsquigarrow X$, and $Q_0 \rightsquigarrow Y$
5 $s_1 := P$, $s_2 := P$
6 **repeat**
7   guess $a, b \in \Sigma(X) \cap \Sigma(Y)$
8     $s_1 := \delta(s_1, a)$, $s_2 := \delta(s_2, b)$
9 **until** $s_1 = X$ *and* $s_2 = Y$
10 Guess states X', Y' of $\mathcal{A}'$ s. t. $X' \cap F \neq \emptyset$ and $Y' \cap F = \emptyset$; `// Non-equiv. check of` X `and` Y
11 $s_1 := X$, $s_2 := Y$
12 **repeat**
13   guess $a \in \Sigma$
14     $s_1 := \delta(s_1, a)$, $s_2 := \delta(s_2, a)$
15 **until** $s_1 = X'$ *and* $s_2 = Y'$
16 **return** `true`

In line 1 the algorithm guesses two states, X and Y, of $\mathcal{A}'$ that are verified to be reachable and in a cycle in line 2. If so, it is verified in lines 10–15 that the states X and Y are not equivalent in $\mathcal{A}'$. If there is no nontrivial cycle in $\mathcal{A}'$ or the guess in line 1 fails, property (2) of Proposition 1 is verified in lines 3–9, and the guessed states X and Y are checked to be non-equivalent in lines 10–15. Notice that in lines 6–9, the algorithm verifies that the states X and Y are reachable from a state P by paths of the same length rather than by paths of different lengths. This is not a problem because line 7 considers only symbols from $\Sigma(X) \cap \Sigma(Y)$. If $\mathcal{A}'$ reaches X under $\Sigma(X) \cap \Sigma(Y)$, it stays in X under those symbols (and analogously for Y). Thus, under $\Sigma(X) \cap \Sigma(Y)$, the states X and Y are reachable from state P by paths of different lengths if and only if they are reachable by paths of the same length. The algorithm is in NPSPACE = PSPACE [35] and returns a positive answer if and only if $\mathcal{A}$ does not accept a piecewise testable language. Since PSPACE is closed under complement [19, 42], piecewise testability is in PSPACE.

PSPACE-hardness follows from a result by Hunt III and Rosenkrantz [18], who have shown that a property $\mathbb{P}$ of languages over the alphabet $\{0,1\}$ such that (i) $\mathbb{P}(\{0,1\}^*)$ is true and (ii) there exists a regular language that is not expressible as a quotient $x \backslash L = \{w \mid xw \in L\}$, for some L for which $\mathbb{P}(L)$ is true, is as hard as to decide "$= \{0,1\}^*$". Since piecewise testability is such a property (piecewise testable languages are closed under quotient) and universality is PSPACE-hard for NFAs, the result implies that piecewise testability for NFAs is PSPACE-hard. □

4. Separability of Regular Languages by Piecewise Testable Languages

We now show that separability of regular languages by piecewise testable languages is PTIME-complete. Since the membership in PTIME is known [8, 30], we prove PTIME-hardness by constructing a log-space reduction from the PTIME-complete monotone circuit value problem [15].

The *monotone circuit value problem* consists of a set of boolean variables $g_1, g_2, \ldots, g_n$ called *gates*, whose values are defined recursively by equalities of the forms $g_i = \mathbf{0}$ (then g_i is called a $\mathbf{0}$-gate), $g_i = \mathbf{1}$ ($\mathbf{1}$-gate), $g_i = g_j \wedge g_k$ ($\wedge$-gate), or $g_i = g_j \vee g_k$ ($\vee$-gate), where $j, k < i$. Here $\mathbf{0}$ and $\mathbf{1}$ are symbols representing the boolean values. The aim is to compute the value of g_n.

A word $a_1 a_2 \cdots a_n$ with $a_i \in \Sigma$ is a subsequence of a word w if $w \in \Sigma^* a_1 \Sigma^* a_2 \Sigma^* \cdots \Sigma^* a_n \Sigma^*$. For languages K and L, a sequence $(w_i)_{i=1}^r$ of words is a *tower* between K and L if $w_1 \in K \cup L$ and, for all $i = 1, 2, \ldots, r-1$, w_i is a subsequence of w_{i+1}, $w_i \in K$ implies $w_{i+1} \in L$, and $w_i \in L$ implies $w_{i+1} \in K$. The number of words in the sequence is the *height* of the tower; the height may be infinite. Languages K and L are not required to be disjoint, but a $w \in K \cap L$ implies an infinite tower $w, w, \ldots$ between K and L.

Our proof is based on the fact that non-separability of languages K and L by a piecewise testable language is equivalent to the existence of an infinite tower between the languages K and L [8].

Theorem 3. *Deciding separability of regular languages represented as NFAs by piecewise testable languages is* PTIME*-complete. It remains* PTIME*-hard even for minimal DFAs.*

Proof. The membership in PTIME was independently shown by Czerwiński et al. [8] and Place et al. [30].

We prove PTIME-hardness by reduction from the monotone circuit value problem (MCVP). Given an instance $g_1, g_2, \ldots, g_n$ of MCVP, we construct two minimal DFAs $\mathcal{A}$ and $\mathcal{B}$ using a log-space reduction and prove that there exists an infinite tower between their languages if and only if the circuit evaluates gate g_n to $\mathbf{1}$. The theorem then follows from the fact that non-separability of two regular languages by a piecewise testable language is equivalent to the existence of an infinite tower [8].

Let $f(i)$ be the element of $\{\wedge, \vee, \mathbf{0}, \mathbf{1}\}$ such that g_i is an $f(i)$-gate. For every $\wedge$-gate and $\vee$-gate, we set $\ell(i)$ and $r(i)$ to be the indices such that $g_i = g_{\ell(i)} f(i) g_{r(i)}$ is the defining equality of g_i. If g_i is a $\mathbf{0}$-gate, we set $f(i) = \ell(i) = r(i) = \mathbf{0}$, and if g_i is a $\mathbf{1}$-gate, we set $f(i) = \ell(i) = r(i) = \mathbf{1}$.

We first construct an automaton $\mathcal{A}' = (Q_{\mathcal{A}'}, \Sigma, \delta_{\mathcal{A}'}, s, F_{\mathcal{A}'})$ with states $Q_{\mathcal{A}'} = \{s, \mathbf{0}, \mathbf{1}, 1, 2, \ldots, n\}$, the input alphabet $\Sigma = \{x, y\} \cup \{a_i, b_i \mid i = 1, \ldots, n\}$, and accepting states $F_{\mathcal{A}'} = \{\mathbf{0}, \mathbf{1}\}$. The initial state of $\mathcal{A}'$ is s and the transition function $\delta_{\mathcal{A}'}$ is defined by $\delta_{\mathcal{A}'}(i, a_i) = \ell(i)$ and $\delta_{\mathcal{A}'}(i, b_i) = r(i)$. In addition, there are two special transitions $\delta_{\mathcal{A}'}(s, x) = n$ and $\delta_{\mathcal{A}'}(\mathbf{1}, y) = s$.

To construct automaton $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, \delta_{\mathcal{B}}, q, F_{\mathcal{B}})$, let $Q_{\mathcal{B}} = \{q, t\} \cup \{i \mid f(i) = \wedge\}$ and $F_{\mathcal{B}} = \{q\}$, where q is also the initial state of $\mathcal{B}$. If $f(i) = \vee$ or $f(i) = \mathbf{1}$, we define $\delta_{\mathcal{B}}(t, a_i) = \delta_{\mathcal{B}}(t, b_i) = t$. If $f(i) = \wedge$, we define $\delta_{\mathcal{B}}(t, a_i) = i$ and $\delta_{\mathcal{B}}(i, b_i) = t$. Finally, we define $\delta_{\mathcal{B}}(q, x) = t$ and $\delta_{\mathcal{B}}(t, y) = q$.

All undefined transitions go to the unique sink states of the respective automata. The automata $\mathcal{A}'$ and $\mathcal{B}$ can be constructed from $g_1, \ldots, g_n$ in logarithmic space. An example of the construction for the circuit $g_1 = \mathbf{0}$, $g_2 = \mathbf{1}$, $g_3 = g_1 \wedge g_2$, $g_4 = g_3 \vee g_3$ is illustrated in Figure 1.

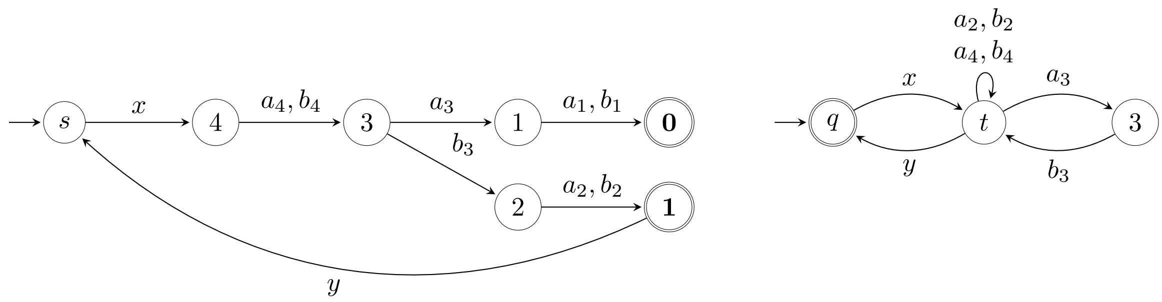

Figure 1: Automata $\mathcal{A}'$ and $\mathcal{B}$ for the circuit $g_1 = \mathbf{0}$, $g_2 = \mathbf{1}$, $g_3 = g_1 \wedge g_2$, $g_4 = g_3 \vee g_3$.

The languages $L(\mathcal{A}')$ and $L(\mathcal{B})$ are disjoint, the automata $\mathcal{A}'$ and $\mathcal{B}$ are deterministic, and $\mathcal{B}$ is minimal. However, automaton $\mathcal{A}'$ need not be minimal because the circuit may contain gates that do not contribute to the definition of the value of g_n. We therefore define a minimal deterministic automaton $\mathcal{A}$ by adding new transitions into $\mathcal{A}'$, each under a fresh symbol, from state s to each of the states $1, 2, \ldots, n-1$, from each of the states $1, 2, \ldots, n$ to state $\mathbf{0}$, and from state $\mathbf{0}$ to state $\mathbf{1}$. This can again be done in logarithmic space. No new transition is defined in $\mathcal{B}$.

Since the language of $\mathcal{B}$ is over Σ, the symbols of $\mathcal{A}$ not belonging to Σ have no effect on the existence of an infinite tower between $L(\mathcal{A})$ and $L(\mathcal{B})$. Namely, there exists an infinite tower between the languages $L(\mathcal{A})$ and $L(\mathcal{B})$ if and only if there exists an infinite tower between $L(\mathcal{A}')$ and $L(\mathcal{B})$. It is therefore sufficient to prove that the circuit evaluates gate g_n to $\mathbf{1}$ if and only if there is an infinite tower between the languages $L(\mathcal{A}')$ and $L(\mathcal{B})$.

The intuition behind the construction is that the symbols of an infinite tower with unbounded number of occurrences correspond to gates that evaluate to $\mathbf{1}$ to satisfy g_n, and that the non-existence of an infinite tower implies the existence of a symbol with bounded number of occurrences in $\mathcal{A}'$ that appears in a non-trivial cycle of the form $a_j b_j$ in $\mathcal{B}$. Such a state corresponds to an $\wedge$-gate, g_j, which cannot be satisfied and causes that g_n evaluates to $\mathbf{0}$ (cf. symbol a_3 in Figure 1).

If there are no $\wedge$-gates, g_n is satisfied if and only if state $\mathbf{1}$ is reachable from state n in $\mathcal{A}'$. Let w be a word under which state $\mathbf{1}$ is reachable from state n. Then $xw \in L(\mathcal{A}')$, $xwy \in L(\mathcal{B})$, $xwyxw \in L(\mathcal{A}'), \ldots$ is an infinite tower between $L(\mathcal{A}')$ and $L(\mathcal{B})$. If state $\mathbf{1}$ is not reachable from state n in $\mathcal{A}'$, then the language $L(\mathcal{A}')$ is finite and there is indeed no infinite tower between $L(\mathcal{A}')$ and $L(\mathcal{B})$.

The problem with $\wedge$-gates is how to ensure that both children of an $\wedge$-gate g_j are satisfied. To this aim, we use the nontrivial cycle under $a_j b_j$ in $\mathcal{B}$, which enforces that both a_j and b_j appear in the words of an infinite tower. Speaking intuitively, automata $\mathcal{A}'$ and $\mathcal{B}$ encode the satisfiability check of g_j (see state g_3 in Figure 1) in the following way. Automaton $\mathcal{A}'$ checks reachability of state $\mathbf{1}$ from state j under a word in $a_j\Sigma^* \cup b_j\Sigma^*$ and automaton $\mathcal{B}$ ensures that a_j appears in a word in $L(\mathcal{B})$ if and only if b_j does. The main idea now is that if there is an infinite tower $(w_i)_{i=1}^{\infty}$ and a_j appears in a word $w_i \in L(\mathcal{A}')$, then both a_j and b_j appear in $w_{i+1} \in L(\mathcal{B})$. By the construction of $\mathcal{A}'$, symbol x appears between any two occurrences of a_j and b_j, hence $\mathcal{B}$ increases the number of occurrences of a_j and b_j in the words of the tower as the height grows. Since the tower is infinite, the number of their occurrences is unbounded. However, to read an unbounded number of a_j and b_j in $\mathcal{A}'$ requires that there is a path from state j to state $\mathbf{1}$ under a word in $a_j\Sigma^*$ as well as under a word in $b_j\Sigma^*$, which (using inductively the same argument for other $\wedge$-gates) is possible only if g_j is satisfied. In Figure 1, the words of $L(\mathcal{A}')$ contain at most one occurrence of a_3, whereas those of $L(\mathcal{B})$ require unbounded number of occurrences of a_3. Thus, there is no infinite tower between the languages of Figure 1.

We now formally prove that the circuit evaluates gate g_n to $\mathbf{1}$ if and only if there is an infinite tower between the languages $L(\mathcal{A}')$ and $L(\mathcal{B})$. The dependence between the gates $g_1, g_2, \ldots, g_n$ can be depicted as a directed acyclic graph $G = (\{1, 2, \ldots, n\}, E)$, where E is defined as $\delta_{\mathcal{A}'}$ without the labels, multiplicities and states $s, \mathbf{0}, \mathbf{1}$. We say that i is *accessible* from j if there is a path from j to i in G.

(Only if) Assume that g_n is evaluated to $\mathbf{1}$. We construct an alphabet Γ, $\{x, y\} \subseteq \Gamma \subseteq \Sigma$, under which both automata $\mathcal{A}'$ and $\mathcal{B}$ have a cycle containing the initial and an accepting state. These cycles then imply the existence of an infinite tower between the languages $L(\mathcal{A}')$ and $L(\mathcal{B})$. Symbol a_i belongs to Γ if and only if g_i is evaluated to $\mathbf{1}$, i is accessible from n, and either $\ell(i) = \mathbf{1}$ or $g_{\ell(i)}$ is evaluated to $\mathbf{1}$. Similarly, b_i belongs to Γ if and only if i is accessible from n, g_i is evaluated to $\mathbf{1}$, and either $r(i) = \mathbf{1}$ or $g_{r(i)}$ is evaluated to $\mathbf{1}$. It is not hard to observe that each transition labeled by a symbol a_i or b_i from Γ is part of a path from n to $\mathbf{1}$ in $\mathcal{A}'$, hence it appears on a cycle in $\mathcal{A}'$ from the initial state s back to state s through the accepting state $\mathbf{1}$. Moreover, the definition of $\wedge$ implies that $a_i \in \Gamma$ if and only if $b_i \in \Gamma$ for each $i = 1, 2, \ldots, n$ such that $f(i) = \wedge$. Notice that $\mathcal{B}$ has a cycle from q to q labeled by xa_ib_iy for each $i = 1, 2, \ldots, n$ with $f(i) \neq \mathbf{0}$. Therefore, both automata $\mathcal{A}'$ and $\mathcal{B}$ have a cycle over the alphabet Γ containing the initial and accepting states. The existence of an infinite tower follows.

(If) Assume that there exists an infinite tower $(w_i)_{i=1}^{\infty}$ between $L(\mathcal{A}')$ and $L(\mathcal{B})$, and, for the sake of contradiction, assume that g_n is evaluated to $\mathbf{0}$. Note that any path from i to $\mathbf{1}$ in $\mathcal{A}'$, where g_i is evaluated to $\mathbf{0}$, must contain a state corresponding to an $\wedge$-gate that is evaluated to $\mathbf{0}$. In particular, this applies to any path in $\mathcal{A}'$ accepting a word of the infinite tower of length at least $n+2$, since such a path contains a subpath from n to $\mathbf{1}$. Let j denote the smallest positive integer such that $f(j) = \wedge$, gate g_j is evaluated to $\mathbf{0}$, and a_j or b_j is in $\cup_{i=1}^{\infty} \operatorname{alph}(w_i)$. The construction of $\mathcal{B}$ implies that both a_j and b_j are in $\cup_{i=1}^{\infty} \operatorname{alph}(w_i)$ because of the nontrivial cycle a_jb_j. Since g_j is evaluated to $\mathbf{0}$, there exists $c \in \{a, b\}$ such that the transition from j under c_j leads to a state σ, where either $\sigma = \mathbf{0}$ or $\sigma < j$ and g_σ is evaluated to $\mathbf{0}$. Consider a word $w_i \in L(\mathcal{A}')$ of the infinite tower containing c_j. If w_i is accepted in $\mathbf{1}$, then the accepting path contains a subpath from σ to $\mathbf{1}$, which yields a contradiction with the minimality of j. Therefore, w_i is accepted in $\mathbf{0}$. However, no symbol of a transition to state $\mathbf{0}$ appears in a word accepted by $\mathcal{B}$ (cf. the symbols a_1 and b_1 in Figure 1), a contradiction again. $\square$

Acknowledgements

The author is very grateful to Štěpán Holub for his comments on the preliminary version of this work, and to an anonymous reviewer for valuable comments. The research was supported by the German Research Foundation (DFG) in Emmy Noether grant KR 4381/1-1 (DIAMOND).

References

[1] Almeida, J., Bartoňová, J., Klíma, O., Kunc, M., 2015. On decidability of intermediate levels of concatenation hierarchies. In: Developments in Language Theory. Vol. 9168 of LNCS. Springer, pp. 58–70.

[2] Almeida, J., Costa, J. C., Zeitoun, M., 2008. Pointlike sets with respect to $\mathcal{R}$ and $\mathcal{J}$. Journal of Pure and Applied Algebra 212 (3), 486–499.

[3] Almeida, J., Zeitoun, M., 1997. The pseudovariety $\mathcal{J}$ is hyperdecidable. RAIRO – Theoretical Informatics and Applications 31 (5), 457–482.

[4] Bojanczyk, M., Segoufin, L., Straubing, H., 2012. Piecewise testable tree languages. Logical Methods in Computer Science 8 (3).

[5] Brzozowski, J. A., Knast, R., 1978. The dot-depth hierarchy of star-free languages is infinite. Journal of Computer and System Sciences 16 (1), 37–55.

[6] Cho, S., Huynh, D. T., 1991. Finite-automaton aperiodicity is PSPACE-complete. Theoretical Computer Science 88 (1), 99–116.

[7] Cohen, R. S., Brzozowski, J. A., 1971. Dot-depth of star-free events. Journal of Computer and System Sciences 5 (1), 1–16.

[8] Czerwiński, W., Martens, W., Masopust, T., 2013. Efficient separability of regular languages by subsequences and suffixes. In: International Colloquium on Automata, Languages and Programming. Vol. 7966 of LNCS. Springer, pp. 150–161.

[9] Czerwiński, W., Martens, W., van Rooijen, L., Zeitoun, M., 2015. A note on decidable separability by piecewise testable languages. In: International Symposium on Fundamentals of Computation Theory. Vol. 9210 of LNCS. Springer, pp. 173–185, and its extended version https://arxiv.org/abs/1410.1042 with G. Zetzsche.

[10] Diekert, V., Gastin, P., Kufleitner, M., 2008. A survey on small fragments of first-order logic over finite words. International Journal of Foundations of Computer Science 19 (3), 513–548.

[11] Fu, J., Heinz, J., Tanner, H. G., 2011. An algebraic characterization of strictly piecewise languages. In: Theory and Applications of Models of Computation. Vol. 6648 of LNCS. Springer, pp. 252–263.

[12] García, P., Ruiz, J., 2004. Learning k-testable and k-piecewise testable languages from positive data. Grammars 7, 125–140.

[13] García, P., Vidal, E., 1990. Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (9), 920–925.

[14] Goubault-Larrecq, J., Schmitz, S., 2016. Deciding piecewise testable separability for regular tree languages. In: International Colloquium on Automata, Languages, and Programming. Vol. 55 of LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 97:1–97:15.

[15] Greenlaw, R., Hoover, H. J., Ruzzo, W. L., 1995. Limits to Parallel Computation: P-Completeness Theory. Oxford University Press.

[16] Hofman, P., Martens, W., 2015. Separability by short subsequences and subwords. In: International Conference on Database Theory. Vol. 31 of LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 230–246.

[17] Hunt III, H. B., 1982. On the decidability of grammar problems. Journal of the ACM 29 (2), 429–447.

[18] Hunt III, H. B., Rosenkrantz, D. J., 1978. Computational parallels between the regular and context-free languages. SIAM Journal on Computing 7 (1), 99–114.

[19] Immerman, N., 1988. Nondeterministic space is closed under complementation. SIAM Journal on Computing 17, 935–938.

[20] Klíma, O., Kunc, M., Polák, L., 2014. Deciding k-piecewise testability, manuscript.

[21] Klíma, O., Polák, L., 2013. Alternative automata characterization of piecewise testable languages. In: Developments in Language Theory. Vol. 7907 of LNCS. Springer, pp. 289–300.

[22] Kontorovich, L., Cortes, C., Mohri, M., 2008. Kernel methods for learning languages. Theoretical Computer Science 405 (3), 223–236.

[23] Kufleitner, M., Lauser, A., 2012. Around dot-depth one. International Journal of Foundations of Computer Science 23 (6), 1323–1340.

[24] Martens, W., Neven, F., Niewerth, M., Schwentick, T., 2015. Bonxai: Combining the simplicity of DTD with the expressiveness of XML schema. In: Principles of Database Systems. pp. 145–156.

[25] Masopust, T., 2016. Piecewise testable languages and nondeterministic automata. In: Mathematical Foundations of Computer Science. Vol. 58 of LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 67:1–67:14.

[26] Masopust, T., Thomazo, M., 2017. On boolean combinations forming piecewise testable languages. Theoretical Computer Science 682, 165–179.

[27] McNaughton, R., Papert, S. A., 1971. Counter-Free Automata. The MIT Press.

[28] Perrin, D., Pin, J.-E., 2004. Infinite words: Automata, semigroups, logic and games. Vol. 141 of Pure and Applied Mathematics. Elsevier, pp. 133–185.

[29] Place, T., 2015. Separating regular languages with two quantifiers alternations. In: ACM/IEEE Symposium on Logic in Computer Science. IEEE Computer Society, pp. 202–213.

[30] Place, T., van Rooijen, L., Zeitoun, M., 2013. Separating regular languages by piecewise testable and unambiguous languages. In: Mathematical Foundations of Computer Science. Vol. 8087 of LNCS. Springer, pp. 729–740.

[31] Place, T., Zeitoun, M., 2014. Going higher in the first-order quantifier alternation hierarchy on words. In: International Colloquium on Automata, Languages and Programming. Vol. 8573 of LNCS. Springer, pp. 342–353.

[32] Place, T., Zeitoun, M., 2015. Separation and the successor relation. In: Symposium on Theoretical Aspects of Computer Science. Vol. 30 of LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 662–675.

[33] Rogers, J., Heinz, J., Bailey, G., Edlefsen, M., Visscher, M., Wellcome, D., Wibel, S., 2010. On languages piecewise testable in the strict sense. In: The Mathematics of Language. Vol. 6149 of LNAI. Springer, pp. 255–265.

[34] Rogers, J., Heinz, J., Fero, M., Hurst, J., Lambert, D., Wibel, S., 2013. Cognitive and sub-regular complexity. In: Formal Grammar. Vol. 8036 of LNCS. Springer, pp. 90–108.

[35] Savitch, W. J., 1970. Relationships between nondeterministic and deterministic tape complexities. Journal of Computer and System Sciences 4 (2), 177–192.

[36] Simon, I., 1972. Hierarchies of events with dot-depth one. Ph.D. thesis, University of Waterloo, Canada.

[37] Simon, I., 1975. Piecewise testable events. In: GI Conference on Automata Theory and Formal Languages. Springer, pp. 214–222.

[38] Sipser, M., 2006. Introduction to the theory of computation, 2nd Edition. Thompson Course Technology.

[39] Stern, J., 1985. Complexity of some problems from the theory of automata. Information and Control 66 (3), 163–176.

[40] Straubing, H., 1981. A generalization of the Schützenberger product of finite monoids. Theoretical Computer Science 13, 137–150.

[41] Straubing, H., 1985. Finite semigroup varieties of the form $\mathbf{V}*\mathbf{D}$. Journal of Pure and Applied Algebra 36, 53–94.

[42] Szelepcsényi, R., 1988. The method of forced enumeration for nondeterministic automata. Acta Informatica 26, 279–284.

[43] Thérien, D., 1981. Classification of finite monoids: The language approach. Theoretical Computer Science 14, 195–208.

[44] Trahtman, A. N., 2001. Piecewise and local threshold testability of DFA. In: International Symposium on Fundamentals of Computation Theory. Vol. 2138 of LNCS. Springer, pp. 347–358.

[45] Wagner, K. W., 2004. Leaf language classes. In: Machines, Computations, and Universality. Vol. 3354 of LNCS. Springer, pp. 60–81.