Alternating Towers and Piecewise Testable Separators

Štěpán Holub\textsuperscript{a,1}, Tomáš Masopust\textsuperscript{b,c,2,*}, Michaël Thomazo\textsuperscript{b,3}

\textsuperscript{a}Charles University, Sokolovská 83, 175 86 Praha, Czech Republic
\textsuperscript{b}TU Dresden, Germany
\textsuperscript{c}Institute of Mathematics, Czech Academy of Sciences, Žižkova 22, 616 62 Brno, Czech Republic

Abstract

Two languages are separable by a piecewise testable language if and only if there exists no infinite tower between them. An infinite tower is an infinite sequence of strings alternating between the two languages such that every string is a subsequence (scattered substring) of all the strings that follow. For regular languages represented by nondeterministic finite automata, the existence of an infinite tower is decidable in polynomial time. In this paper, we investigate the complexity of a particular method to compute a piecewise testable separator. We show that it is closely related to the height of maximal finite towers, and provide the upper and lower bounds with respect to the size of the given nondeterministic automata. Specifically, we show that the upper bound is polynomial with respect to the number of states with the cardinality of the alphabet in the exponent. Concerning the lower bound, we show that towers of exponential height with respect to the cardinality of the alphabet exist. Since these towers mostly turn out to be sequences of prefixes, we also provide a comparison with towers of prefixes.

Keywords: Separability, separators, piecewise testable languages, alternating towers, complexity

2010 MSC: 68Q45, 68Q17, 68Q25, 03D05

1. Introduction

The separation problem appears in many disciplines of mathematics and computer science, such as algebra \cite{3, 4, 22}, logic \cite{24, 25}, formal languages \cite{7, 23}, learning theory \cite{16}, and recently also in databases and query answering \cite{8}. The latter topic is our original motivation we have investigated since the work of \cite{8}. In that work, the motivation comes from two practical problems. The first is to make XML Schema human-readable. XML Schema is a schema language for XML that is widely accepted and supported, but is rather machine-readable than human-readable. Of course, it increases the expressiveness of Document Type Definition \cite{5}, but this increase goes hand in hand with loss of simplicity. Recently, the BonXai schema language has been proposed in \cite{19} as an attempt to design a human-readable schema language. The BonXai schema is a set of rules \( L_i \rightarrow R_i \), where \( L_i \) and \( R_i \) are regular expressions. An XML document (unranked tree) belongs to the language of the schema if for every node the labels of its children form a string belonging to \( R_i \) and its ancestors form a string belonging to \( L_i \), see \cite{19} for more details. The problem we faced when translating (the finite automaton embedded in) an XML Schema Definition into an equivalent BonXai schema was that the automatically generated regular expressions \( L_i \) were not human-readable. Therefore, we restricted the considered regular expressions to some “simple” variants, e.g., piecewise testable languages \cite{8}.

The second motivation comes from the observation that regular expressions are used to match paths between nodes in a graph, hence their efficient evaluation is relevant in database and knowledge-base systems. However, the graph (database, knowledge base) can be so huge that the exact evaluation is not feasible in a reasonable time \cite{18, 21}. As a
solution, the expression could be rewritten to (an)other expression(s) that can be evaluated efficiently. For instance, it could be rewritten to two expressions $r_s$ and $r_r$. defining the languages that should and should not be matched in the answer. The question is then whether there exists such a “simple” query, and how to obtain it.

It is not hard to see that the previous problems reduce to the language separation problem. Given two languages $K$ and $L$ and a family of languages $\mathcal{F}$, the 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 with the other. Recently, it has been independently shown in [8] and [23] that the separation problem for two 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 alphabet. It should be noted that an algorithm polynomial with respect to the number of states and exponential with respect to the size of the alphabet has been known in the literature [1, 4]. In [8], the separation problem has been shown to be equivalent to the non-existence of an infinite tower between the languages. Namely, the languages have been shown separable by a piecewise testable language if and only if there does not exist an infinite tower between them. In [23], another technique has been used to prove the polynomial time bound for the decision procedure, and a doubly exponential upper bound on the index of the separating piecewise testable language. This information can be further used to construct a separating piecewise testable language with the complexity, in general, exponential with respect to the index.

However, there exists a simple method (in the meaning of description) to decide the separation problem and to compute a separating piecewise testable language, whose running time corresponds to the height of the highest finite tower. This method is the original work of [5] and is recalled in Section 4. The relationship between the complexity and the height of towers has motivated the study of this paper to investigate the upper and lower bounds on the height of finite towers in the presence of no infinite towers. So far, to the best of our knowledge, the only published result in this direction is a paper by Stern [11], who provided an exponential upper bound $2^{\Omega(N)}$ on the height of alternating towers between a piecewise testable language and its complement, where $N$ denotes the number of states of the minimal deterministic finite automaton.

Our contribution in this paper, which is a major revision and extension of [13], are the upper and lower bounds on the height of maximal finite towers between two regular languages represented by nondeterministic finite automata. Since the existence of towers of arbitrary height implies the existence of an infinite tower [8], we restrict our attention only to the case where no infinite tower exists between the languages. We prove that the upper bound is polynomial with respect to the number of states, but exponential with respect to the size of the alphabet (Theorem 6). Concerning the lower bound, we first improve the previous result showing that the bound is tight for binary regular languages up to a linear factor (Theorem 7). The main result then shows that we can achieve an exponential lower bound for NFAs with respect to the size of the alphabet (Theorems 8 and 9). The lower bound for DFAs is discussed in Theorems 10 and 11. Since our towers for NFAs are in fact sequences of prefixes, we investigate the towers of prefixes in Section 7. We prove tight upper bounds on the height of towers of prefixes in Theorem 13 and Corollary 14. We show that the problem is NL-complete for both NFAs and DFAs (Theorems 17 and Corollary 18). Finally, Section 3 provides an overview of related results. To complete it, we prove that the piecewise-testability problem is PSPACE-complete for NFAs (Theorem 12) and that separability of regular languages (represented by NFAs or DFAs) by piecewise testable languages is P-complete (Theorem 5).

2. Preliminaries

We assume that the reader is familiar with automata and formal language theory [17, 26, 30]. The cardinality of a set $A$ is denoted by $|A|$ and the power set of $A$ by $2^A$. An alphabet $\Sigma$ is a finite nonempty set. The elements of an alphabet are called letters. The free monoid generated by $\Sigma$ is denoted by $\Sigma^*$. A string over $\Sigma$ is any element of $\Sigma^*$. The empty string is denoted by $\epsilon$. For a string $w \in \Sigma^*$, $\text{alph}(w) \subseteq \Sigma$ denotes the set of all letters occurring in $w$, and $|w_i|$ denotes the number of occurrences of letter $a$ in $w$.

**Automata.** A nondeterministic finite automaton (NFA) is a quintuple $M = (Q, \Sigma, \delta, Q_0, F)$, where $Q$ is the finite nonempty set of states, $\Sigma$ is the input alphabet, $Q_0 \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of accepting states, and $\delta : Q \times \Sigma \to 2^Q$ is the transition function. The transition function is extended to the domain $2^Q \times \Sigma^*$ in the usual way. The language accepted by $M$ is the set $L(M) = \{w \in \Sigma^* \mid \delta(Q_0, w) \cap F \neq \emptyset\}$. A path $\pi$ from a state $q_0$ to a state
The automaton states are pairwise distinguishable. The number of states on the longest simple path in $M$ is called the depth of the automaton $M$. We use the notation $q_0 \xrightarrow{a_1 \cdots a_n} q_n$ to denote that there exists a path from $q_0$ to $q_n$ under the string $a_1 a_2 \cdots a_n$. The NFA $M$ has a cycle over an alphabet $\Gamma \subseteq \Sigma$ if there exists a state $q$ and a string $w$ over $\Sigma$ such that $\text{alph}(w) = \Gamma$ and $q \xrightarrow{w} q$.

The NFA $M$ is deterministic (DFA) if $|Q_0| = 1$ and $|\delta(q, a)| = 1$ for every $q$ in $Q$ and $a$ in $\Sigma$. We identify singleton sets with their elements and write $q$ instead of $\{q\}$. The transition function $\delta$ is then a map from $Q \times \Sigma$ to $Q$ that is extended to the domain $Q \times \Sigma^*$ in the usual way. Two states of a DFA are distinguishable if there exists a string $w$ that is accepted from one of them and rejected from the other, otherwise they are equivalent. A DFA is minimal if all its states are reachable and pairwise distinguishable.

In this paper, we assume that all automata under consideration have no useless states, that is, every state appears on an accepting path.

**Embedding.** For two strings $v = a_1 a_2 \cdots a_n$ and $w \in \Sigma^* a_1^* a_2^* \cdots a_n^* \Sigma^*$, we say that $v$ is a subsequence of $w$ or that $v$ can be embedded into $w$, denoted by $v \preceq w$. For two languages $K$ and $L$, we say that the language $K$ can be embedded into the language $L$, denoted by $K \preceq L$, if for every string $w$ in $K$, there exists a string $w'$ in $L$ such that $w \preceq w'$. We say that a string $w$ can be embedded into the language $L$, denoted by $w \preceq L$, if $|w| \preceq L$.

**Towers.** We define (alternating subsequence) towers as a generalization of Stern’s alternating towers, cf. [31]. For two languages $K$ and $L$ and the subsequence relation $\preceq$, we say that a sequence $(w_i)_{i=1}^r$ of strings is an (alternating subsequence) tower between $K$ and $L$ if $w_i \in K \cup L$ and, for all $i = 1, \ldots, r - 1$,

- $w_i \preceq w_{i+1}$,
- $w_i \in K$ implies $w_{i+1} \in L$, and
- $w_i \in L$ implies $w_{i+1} \in K$.

We say that $r$ is the height of the tower. In the same way, we define an infinite sequence of strings to be an infinite (alternating subsequence) tower between $K$ and $L$. If the languages are clear from the context, we usually omit them. Notice that the languages are not required to be disjoint, however, if there exists a $w \in K \cap L$, then there exists an infinite tower, namely $w, w, w, \ldots$.

If we talk about a tower between two automata, we mean the tower between their languages.

**Piecewise testable languages.** A regular language is piecewise testable if it is a finite boolean combination of languages of the form $\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \cdots a_k \Sigma^*$, where $k \geq 0$ and $a_i \in \Sigma$, see [23,29] for more details.

** Separability.** Let $K$ and $L$ be two 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 $K$.

Prefixes and towers of prefixes. We say that a string $v \in \Sigma^*$ is a prefix of a string $w \in \Sigma^*$, denoted by $v \preceq w$, if $w = vu$, for some string $u \in \Sigma^*$. A sequence $(w_i)_{i=1}^r$ of strings is a tower of prefixes between two languages $K$ and $L$ if $w_1 \in K \cup L$ and, for all $i = 1, 2, \ldots, r - 1$, $w_i \preceq w_{i+1}$, $w_i \in K$ implies $w_{i+1} \in L$, and $w_i \in L$ implies $w_{i+1} \in K$.

3. Relevant results

In this section, we first briefly summarize the results concerning piecewise testable languages and separability that are relevant to the topic of this paper.

Piecewise testable languages were studied by Simon in his PhD thesis [28], see also [29]. He proved that piecewise testable languages are exactly those regular languages whose syntactic monoid is $\mathcal{F}$-trivial. He also provided various characterizations of piecewise testable languages in terms of monoids, automata, etc. These languages found applications in algebra [3,4], logic [24,25], formal languages [8,23] and learning theory [10], to mention a few.
The fundamental question was how to efficiently recognize whether a given regular language is piecewise testable. The solution to this problem was provided by Stern in 1985 and improved by Trahtman in 2001. Stern showed that piecewise testability of a regular language represented by a deterministic finite automaton is decidable in polynomial time [32]. He provided an \(O(n^3)\) algorithm, where \(n\) is the number of states. Trahtman [33] improved Stern’s result and obtained a quadratic-time algorithm to decide piecewise testability for deterministic finite automata. In 1991, Cho and Huynh [6] proved that piecewise testability is NL-complete for deterministic finite automata. Although the algorithm is formulated for deterministic finite automata, it can be modified for nondeterministic automata. In 2013, Czerwiński, Martens, Masopust [8] and, independently, Place, Van Rooijen, Zeitoun [23] provided polynomial-time algorithms (with respect to both the size of the state space and the cardinality of the alphabet) to decide separability of regular languages represented by nondeterministic finite automata by piecewise testable languages. In this section, we show that separability of regular languages represented by NFAs or DFAs by piecewise testable languages is P-complete.

It should be mentioned that separability has also been studied for other languages. For instance, separability of context-free languages by regular languages was shown undecidable a long time ago, cf. [14]. In particular, it is shown in [14] that even separability of simple context-free languages (so-called s-languages) to definite languages (a strict subfamily of regular languages) is undecidable.

### 3.1. The piecewise-testability problem for NFAs is PSPACE-complete

The piecewise-testability problem asks whether, given a nondeterministic finite automaton \(A\) over an alphabet \(\Sigma\), the language \(L(A)\) is piecewise testable. Although the containment to PSPACE follows basically from the result by Cho and Huynh [6], we prefer to provide the proof here for two reasons. First, we would like to provide an unfamiliar reader with a method to recognize whether a given regular language is piecewise testable. Second, Cho and Huynh assume that the input is the minimal DFA, hence it is necessary to extend their algorithm with a non-equivalence check.

#### Proposition 1

A regular language \(L\) is not piecewise testable if and only if the minimal DFA \(M\) for \(L\) either (1) contains a nontrivial (non-self-loop) cycle or (2) there are three distinct states \(p, q, q'\) such that there are paths from \(p\) to \(q\) and from \(p\) to \(q'\) in the graph \(G(M, \Sigma(q) \cap \Sigma(q'))\), where \(G(M, \Gamma)\) denotes the transition diagram of the DFA \(M\) restricted to edges labeled by letters from \(\Gamma\), and \(\Sigma(q) = \{a \in \Sigma | q \xrightarrow{a} q\}\).

#### Lemma 2

The piecewise-testability problem for NFAs is in PSPACE.

**Proof.** Let \(A = (Q, \Sigma, \delta, Q_0, F)\) be an NFA. Since the automaton is nondeterministic, we cannot directly use the algorithm by Cho and Huynh [6]. However, we can consider the DFA \(A'\) obtained from \(A\) by the standard subset construction. The states of \(A'\) are subsets of states of \(A\). Now we only need to modify Cho and Huynh’s algorithm to check whether the guessed states are distinguishable.

The entire algorithm is presented as Algorithm 1. In line 1 it guesses two states, \(X\) and \(Y\), of \(A'\) that are verified to be reachable and in a cycle in lines 2-4. If so, it is verified in lines 12-17 that the states \(X\) and \(Y\) are not equivalent in \(A'\). If there is no cycle in \(A'\), property (2) of the proposition is verified in lines 5-11, and the guessed states \(X\) and \(Y\) are verified to be non-equivalent in lines 12-17.

The algorithm is in \(\text{NPSPACE} = \text{PSPACE} [27]\) and returns a positive answer if and only if \(A\) does not accept a piecewise testable language. Since PSPACE is closed under complement, piecewise testability is in PSPACE. \(\square\)

#### Lemma 3

The piecewise-testability problem for NFAs is PSPACE-hard.

**Proof.** We prove PSPACE-hardness by reduction from the universality problem, which is PSPACE-complete [10]. The universality problem asks whether, given an NFA \(A\) over \(\Sigma\), the language \(L(A) = \Sigma^*\).
### Algorithm 1 Non-piecewise testability (symbol \(\rightsquigarrow\) stands for reachability)

1: Guess states \(X, Y \subseteq Q\) of \(A'\): \(\triangleright \) Verify property (1)
2: if \(Q_0 \rightsquigarrow X \rightsquigarrow Y \rightsquigarrow X\) then
3: \(\text{go to line } 12\)
4: end if
5: Guess states \(P, X, Y \subseteq Q\) of \(A'\): \(\triangleright \) Verify property (2)
6: Check \(Q_0 \rightsquigarrow P, Q_0 \rightsquigarrow X,\) and \(Q_0 \rightsquigarrow Y\):
7: \(s_1 := P;\) \(\quad s_2 := P;\)
8: repeat guess \(a, b \in \Sigma \cap \Sigma(Y)\);
9: \(s_1 := \delta(s_1, a);\)
10: \(s_2 := \delta(s_2, b);\)
11: until \(s_1 = X\) and \(s_2 = Y;\)
12: Guess states \(X', Y' \subseteq Q\) of \(A'\) such that \(X' \cap F \neq \emptyset\) and \(Y' \cap F = \emptyset;\) \(\triangleright \) Check that \(X\) and \(Y\) are not equivalent
13: \(s_1 := X;\) \(\quad s_2 := Y;\)
14: repeat guess \(a \in \Sigma;\)
15: \(s_1 := \delta(s_1, a);\)
16: \(s_2 := \delta(s_2, a);\)
17: until \(s_1 = X'\) and \(s_2 = Y';\)
18: return \'yes\';

Let \(A\) be an NFA with a single initial state \(q_0\) (this is not a restriction). Check whether \(L(A) = \emptyset\) (in linear time). If so, return the minimal DFA \(A'\) for the non-piecewise testable language \((aa)^\ast\). If \(L(A) \neq \emptyset\), let \(x\) be a new letter, and let \(d\) be a new state. We “complete” the automaton \(A\) in the sense that if no \(a\)-transition is defined in a state \(q\), for \(a\) in \(\Sigma\), we add an \(a\)-transition from \(q\) to \(d\). State \(d\) contains self-loops under all letters of \(\Sigma\), but not under \(x\). Now, we add an \(x\)-transition from each state, including \(d\), to the initial state \(q_0\). Let \(A'\) denote the resulting automaton.

If \(L(A) = \Sigma\), we show that the language \(L(A')\) is piecewise testable by showing that \(L(A') = (\Sigma \cup \{x\})^\ast\). Indeed, \(L(A) \subseteq L(A')\), so it remains to show that every string containing \(x\) is accepted by \(A'\). However, let \(w = w_1 x w_2\), where \(w_1 \in (\Sigma \cup \{x\})^\ast\) and \(w_2 \in \Sigma^\ast\). By the construction, \(w_1 x\) leads the automaton back to the initial state, and \(w_2\) leads the automaton to an accepting state, because \(w_2 \in L(A) = \Sigma^\ast\). Thus, \(w \in L(A')\).

Assume that \(L(A) \neq \Sigma\). If \(L(A) = \emptyset\), then \(L(A') = (aa)^\ast\), which is not piecewise testable. If \(L(A) \neq \emptyset\), consider the minimal DFA \(A''\) computed from \(A'\) by the standard subset construction and minimization. The DFA has at least two states, otherwise its language is either universal or empty. Every state of \(A''\) is a nonempty subset of states of \(A\) (actually it is an equivalence class of such subsets, but we pick one as a representative). The empty set is not reachable because \(A'\) is complete. Let \(X \neq \{q_0\}\) be a state of \(A''\). Then \(X\) is reachable from the initial state \(\{q_0\}\), and goes back to \(\{q_0\}\) under \(x\), which means that there is a cycle in the minimal DFA. By (1) of Proposition 1, the language is not piecewise testable.

We have proved the following result.

**Theorem 4.** The piecewise-testability problem for NFAs is PSPACE-complete.

### 3.2. The separation problem is P-complete

The separation problem of two regular languages by a piecewise testable language is known to be equivalent to the non-existence of an infinite tower [8] and is decidable in polynomial time [8][23]. In this section, we show that the problem is P-complete.

The monotone circuit value problem (MCVP) is composed of a set of boolean variables (usually called “gates”) \(g_1, g_2, \ldots, g_n\), whose values are defined recursively by equalities of the forms \(g_i = 0\) (then \(g_i\) is called a 0-gate), \(g_i = 1\) (1-gate), \(g_i = g_j \land g_k\) (\&-gate), or \(g_i = g_j \lor g_k\) (\lor-gate), where \(j, k < i\). Here \(0\) and \(1\) are symbols representing the boolean values. The aim is to compute the value of \(g_n\). Let \(f(i)\) be the element of \(\{\land, \lor, 0, 1\}\) such that \(g_i\) is an \(f(i)\)-gate. For every \&-gate and \lor-gate, we set \(\ell(i)\) and \(r(i)\) to be the indices such that \(g_i = g_{\ell(i)} \land g_{r(i)}\) is the defining equality of \(g_i\). For a 0-gate \(g_i\), we set \(f(i) = \ell(i) = r(i) = 0\), and we set \(f(i) = \ell(i) = r(i) = 1\) if \(g_i\) is a 1-gate.

5
Theorem 5. The separation problem of two regular languages represented by NFAs by piecewise testable languages is P-complete. It is P-complete even for regular languages represented by minimal DFAs.

Proof. The problem was shown to be in P in [8, 23]. Thus, it remains to prove P-hardness.

Given an instance \( g_1, g_2, \ldots, g_n \) of MCVP, we construct two minimal deterministic finite automata \( A \) and \( B \) using a log-space reduction and prove that there exists an infinite tower between their languages if and only if the circuit evaluates \( g_n \) to 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].

We first construct an automaton \( A' \). Let \( Q_A = \{ s, 0, 1, 1, 2, \ldots, n \} \) and \( F_A = \{ 0, 1 \} \). The initial state of \( A' \) is \( s \) and the transition function \( \delta_A \) is defined by \( \delta_A(i, a_i) = \ell(i) \) and \( \delta_A(i, b_i) = r(i) \). In addition, there are two special transitions \( \delta_A(s, x) = n \) and \( \delta_A(1, y) = s \). To construct the automaton \( B \), let \( Q_B = \{ q, t \} \cup \{ i \ | \ f(i) = \land \} \) and \( F_B = \{ q \} \), where \( q \) is also the initial state of \( B \). If \( f(i) = \lor \) or \( f(i) = 1 \), then \( \delta_B(t, a_i) = \delta_B(t, b_i) = t \). If \( f(i) = \land \), then \( \delta_B(t, a_i) = i \) and \( \delta_B(t, b_i) = t \). Finally, we define \( \delta_B(q, x) = t \) and \( \delta_B(q, y) = q \). (As usual, all undefined transitions go to the unique sink states of the respective automata.) An example of this construction for the circuit \( g_1 = 0, g_2 = 1, g_3 = g_1 \land g_2, g_4 = g_3 \lor g_3 \) is shown in Figure 1.

The languages \( L(A') \) and \( L(B) \) are disjoint, the automata \( A' \) and \( B \) are deterministic, and \( B \) is minimal. However, the automaton \( A' \) need not be minimal, since 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 \( A \) by adding transitions into \( A' \), each under a fresh letter, from \( s \) to each of \( 1, 2, \ldots, n - 1 \), from each of \( 1, 2, \ldots, n \) to state 0, and from 0 to 1. No new transition is defined in \( B \).

By construction, there exists an infinite tower between the languages \( L(A) \) and \( L(B) \) if and only if there exists an infinite tower between \( L(A') \) and \( L(B) \). It is therefore sufficient to prove that the circuit evaluates \( g_n \) to 1 if and only if there is an infinite tower between the languages \( L(A') \) and \( L(B) \).

(Only if) Assume that \( g_n \) is evaluated to 1. Let \( \{ x, y \} \subseteq \Sigma \subseteq \{ a_i, b_i \ | \ i = 1, 2, \ldots, n \} \cup \{ x, y \} \) be the alphabet defined as follows. The letter \( a_i \) is an element of \( \Sigma \) if and only if \( g_i \) is evaluated to 1, \( i \) is accessible from \( n \), and either \( \ell(i) = 1 \) or \( g(i) \) is evaluated to 1. Similarly, we have \( b_i \in \Sigma \) if and only if \( i \) is accessible from \( n \) and \( g_i \) is evaluated to 1, and either \( r(i) = 1 \) or \( g(i) \) is evaluated to 1. It is not hard to observe that each transition labeled by a letter \( a_i \) or \( b_i \) from \( \Sigma \) is part of a path in \( A' \) from \( n \) to 1. Moreover, the definition of \( \land \) implies that \( a_i \in \Sigma \) if and only if \( b_i \in \Sigma \) for each \( i = 1, 2, \ldots, n \) such that \( f(i) = \land \). Notice that \( B \) has a cycle from \( q \) to \( q \) labeled by \( x_{a,b} x_{a,b} y_{a,b} \) for each \( i = 1, 2, \ldots, n \) such that \( f(i) = \lor \) and also a cycle from \( q \) to \( q \) labeled by \( x_{a,b} y_{a,b} \) for each \( i = 1, 2, \ldots, n \) such that \( f(i) = \lor \). Therefore, both automata \( A' \) and \( B \) have a cycle over the alphabet \( \Sigma \) 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 \( A' \) and \( B \) and, for the sake of contradiction, assume that \( g_n \) is evaluated to 0. Note that any path from \( i \) to 1, where \( g_i \) is evaluated to 0, must contain a state corresponding to an \( \land \)-gate that is evaluated to 0. In particular, this applies to any path in \( A' \) accepting some \( w_i \), of length at least \( n + 2 \), since such a path contains a subpath from \( n \) to 1. Thus, let \( j \) be the smallest positive integer such that \( f(j) = \land \), gate \( g_j \) is evaluated to 0, and \( a_j \) or \( b_j \) is in \( \cup_{i=1}^n \text{alp}(w_i) \). The construction of \( B \) implies that both \( a_j \) and \( b_j \) are in \( \cup_{i=1}^n \text{alp}(w_i) \). Since \( g_j \) is evaluated to 0, there exists \( c \in \{ a, b \} \) such that the transition from \( j \) under \( c_j \) leads to a state \( \sigma \), where either \( \sigma = 0 \) or \( \sigma < j \) and \( g_x \) is evaluated to 0. Consider a string \( w_i \in L(A') \) containing \( c_j \). If \( w_i \) is accepted in 1, then the
accepting path contains a subpath from \( \sigma \) to \( \mathbf{1} \), which yields a contradiction with the minimality of \( j \). Therefore, \( w_j \) is accepted in \( \mathbf{0} \). However, no letter of a transition to state \( \mathbf{0} \) appears in a string accepted by \( B \), a contradiction again. \( \square \)

4. Computing a piecewise testable separator

We now recall the simple method \([2] \) to decide the separation problem and to compute a separating piecewise testable language, and show that its running time corresponds to the height of the highest finite tower.

Let \( L_0 \) and \( R_0 \) be two disjoint regular languages represented by NFAs. The method first constructs a decreasing sequence of languages \( \cdots \subseteq R_2 \subseteq L_2 \subseteq R_1 \subseteq L_1 \subseteq R_0 \) such that there exists a piecewise testable separator if and only if from some point on all the languages are empty. Then, the nonempty languages of the sequence are used to construct a piecewise testable separator.

For \( k \geq 1 \), let \( L_k = \{ w \in L_{k-1} \mid \text{w is not embeddable} \} \) denote the set of all strings of \( L_{k-1} \) that can be embedded into \( L_{k-1} \), and let \( R_k = \{ w \in R_{k-1} \mid \text{w is not embeddable} \} \), see Figure 2. Let \( K \) be a language accepted by an NFA \( A = (Q, \Sigma, \delta, Q_0, F) \). Let \( \downarrow \) denote the language of all subsequences of the language \( K \). It is accepted by the NFA \( A_{\downarrow} = (Q, \Sigma, \delta', Q_0, F) \), where \( \delta'(q, a) = \delta(q, a) \) and \( \delta'(q, \varepsilon) = \bigcup_{w \in \mathcal{L}} \delta(q, a) \). Then \( L_k = L_{k-1} \setminus \downarrow \) and \( R_k = R_{k-1} \setminus \downarrow \), and analogously for \( R_k \), thus the constructed languages are regular.

We now show that there exists a constant \( B \geq 1 \) such that \( L_B = L_{B+1} = \ldots \), which also implies \( R_B = R_{B+1} = \ldots \). Assume that no such constant exists. Then there are infinitely many strings \( v_i \in L_\ell \setminus L_{\ell+1} \) for all \( \ell \geq 1 \), as depicted in Figure 2. By Higman's lemma \([12] \), there exist \( i < j \) such that \( v_i \not\subseteq v_j \), hence \( v_i \not\subseteq R_{j-1} \), which is a contradiction because \( v_j \not\subseteq R_i \) and \( R_{j-1} \subseteq R_i \).

By construction, languages \( L_B \) and \( R_B \) are mutually embeddable into each other, \( L_B \subseteq R_B \subseteq L_B \), which describes a way how to construct an infinite tower. Thus, if there is no infinite tower, languages \( L_B \) and \( R_B \) must be empty.

The constant \( B \) depends on the height of the highest finite tower. Let \( \{ w_j \}_{j=1}^{\ell} \) be a maximal finite tower between \( L_0 \) and \( R_0 \) and assume that \( w_\ell \) belongs to \( L_0 \). In the first step, the method eliminates all strings that cannot be embedded into \( L_0 \), hence \( w_\ell \) does not belong to \( L_1 \), but \( (w_j)_{j=1}^{\ell} \) is a tower between \( L_1 \) and \( R_0 \). Thus, in each step of the algorithm, all maximal strings of all finite towers (belonging to the language under consideration) are eliminated, while the rests of towers still form towers between the resulting languages. Therefore, as long as there is a maximal finite tower, the algorithm can make another step.

Assume that there is no infinite tower (that is, \( L_B = R_B = \emptyset \)). We use the languages computed above to construct a piecewise testable separator. For a string \( w = a_1a_2\cdots a_t \), we define \( L_w = \Sigma a_1\Sigma' a_2\Sigma' \cdots \Sigma a_t\Sigma' \), which is piecewise testable by definition. Let \( up(L) = \bigcup_{w \in L} L_w \) denote the language of all supersequences. The language \( up(L) \) is regular and its NFA is constructed from an NFA for \( L \) by adding self-loops under all letters to all states, cf. \([15] \) for more details. By Higman's Lemma \([12] \), \( up(L) \) can be written as a finite union of languages of the form \( L_w \), for some \( w \in L \), hence it is piecewise testable. For \( k = 1, 2, \ldots, B \), we define the piecewise testable languages

\[
S_k = up(R_0 \setminus R_k) \cup up(L_0 \setminus L_k)
\]

and show that \( S = \bigcup_{k=1}^B S_k \) is a piecewise testable separator of \( L_0 \) and \( R_0 \).

More specifically, we show that \( L_0 \cap S_k = \emptyset \) and that \( R_0 \subseteq S \). To prove the former, let \( w \in L_0 \). If \( w \in L_0 \setminus L_k \), then \( w \notin up(L_0 \setminus L_k) \), hence \( w \notin S_k \). If \( w \in L_k \) and \( w \notin up(R_0 \setminus R_k) \), then there exists \( v \in R_0 \setminus R_k \) such that \( v \subseteq w \). However, \( R_k = \{ u \in R_0 \mid u \subseteq L_k \} \), hence \( v \subseteq R_k \), which is a contradiction proving that \( L_0 \cap S_k = \emptyset \). To prove the latter, we show that \( R_{k-1} \setminus R_k \subseteq S_k \). Then also \( R_0 = \bigcup_{k=1}^B (R_{k-1} \setminus R_k) \subseteq S \). To show this, notice that \( R_{k-1} \setminus R_k \subseteq R_0 \setminus R_k \subseteq up(R_0 \setminus R_k) \).
If \( w \in R_{k-1} \) and \( w \in \text{up}(L_1 \setminus L_k) \), then there exists \( v \in L_0 \setminus L_k \) such that \( v \leq w \). However, \( L_k = \{ u \in L_0 \mid u \not\in R_{k-1} \} \), hence \( v \in L_k \), a contradiction. Thus, we have shown that \( L_0 \cap S = \emptyset \) and \( R_0 \subseteq S \). Moreover, \( S \) is piecewise testable because it is a finite boolean combination of piecewise testable languages.

Recall that we have shown above that the complexity of this method is closely related to the height of the highest finite tower. Therefore, in the rest of this paper, we focus on this problem. However, we would like to point out several directions of the future investigation here.

Consider the sequence of languages \( R_0, L_1, R_1, \ldots, L_B, R_B \) constructed by the method. The size of the representation of the separator \( S \) depends not only on the number of languages in the sequence, but also on their representation. Let \( Z_k \) denote the \( k \)th language in the sequence. If the languages are represented by NFAs, the method gives the bound \( O(n^m) \) on the size of the NFA for the language \( Z_k \), where \( n \) is the number of states of the NFAs for \( L_0 \) and \( R_0 \). Another option would be to use alternating finite automata (AFAs) for obvious reasons in their complexities for boolean operations. However, the problem now comes from the operation \( \text{down}(Z_k) \). Even if we were able to obtain a bound \( \Omega(c^d \cdot n) \) on the size of the AFA representation of the language \( Z_k \), for some constant \( c \), the only interesting case would be the case of \( c = 1 \); otherwise, the problem would still be exponential. It turns out that it is not the case. It has recently been shown in \([15]\) that the downward closure operation \( \text{down}() \) for AFAs can actually require an exponential blow-up of states.

Finally, even though our study in the rest of the paper shows that these representations can be doubly exponential with respect to the size of the input NFAs, practical experiments show that at some point the size of the automata does not grow anymore, but actually decreases. Indeed, the languages \( L_B = R_B \) are empty, hence a representation of size \( \Omega(n^{2B}) \) is not needed for them.

### 5. The upper bound on the height of towers

It was shown in \([8]\) that there exists either an infinite tower or a constant bound on the height of any tower. We now establish an upper bound on that constant in terms of the number of states, \( n \), and the cardinality of the alphabet, \( m \). The bound given by Stern for minimal DFAs is \( 2^{m \cdot n} \). Our new bound is \( O(n^m) = O(2^{m \log n}) \) and holds for NFAs. Thus, our bound is polynomial with respect to the number of states if the alphabet is fixed, but exponential if the alphabet grows. For \( m < n / \log n \), the bound is less than \( 2^n \), but we show later that if \( m = O(n) \), the height of towers may grow exponentially with respect to the number of states.

Before we state the theorem, we recall that the depth of an automaton is the number of states on the longest simple path, thus it is bounded by the number of states of the automaton.

#### Theorem 6

Let \( A_0 \) and \( A_1 \) be NFAs over an alphabet \( \Sigma \) of cardinality \( m \) with depth at most \( n \). Assume that there is no infinite tower between the languages \( L(A_0) \) and \( L(A_1) \). Let \( (w_i)_{i=1}^n \) be a tower between \( L(A_0) \) and \( L(A_1) \) such that \( w_i \in L(A_{i \mod 2}) \). Then \( r \leq \frac{m^{m-1}}{m-1} \).

**Proof.** First, we define some new concepts. We say that \( w = v_1 v_2 \cdots v_k \) is a **cyclic factorization** of \( w \) with respect to a pair of states \( (q, q') \) in an automaton \( A \), if there is a sequence of states \( q_0, \ldots, q_{k-1}, q_k \) such that \( q_0 = q \), \( q_k = q' \), and \( q_{i-1} \overset{v_i}{\rightarrow} q_i \), for each \( i = 1, 2, \ldots, k \), and either \( v_i \) is a letter, or the path \( q_{i-1} \overset{v_i}{\rightarrow} q_i \) contains a cycle over \( \text{alph}(v_i) \). We call \( v_i \) a letter factor if it is a letter and \( q_{i-1} \neq q_i \), and a cycle factor otherwise. The factorization is **trivial** if \( k = 1 \). Note that this factorization is closely related to the one given in \([1]\), see also \([2, \text{Theorem 8.1.11}]\).

We first show that if \( q' \in \delta(q, w) \) in some automaton \( A \) with depth \( n \), then \( w \) has a cyclic factorization \( v_1 v_2 \cdots v_k \) with respect to \( (q, q') \) that contains at most \( n \) cycle factors and at most \( n - 1 \) letter factors. Moreover, if \( w \) does not admit the trivial factorization with respect to \( (q, q') \), then \( \text{alph}(v_i) \) is a strict subset of \( \text{alph}(w) \) for each cycle factor \( v_i \), \( i = 1, 2, \ldots, k \).

Consider a path \( \pi \) of the automaton \( A \) from \( q \) to \( q' \) labeled by a string \( w \). Let \( q_0 = q \) and define the factorization \( w = v_1 v_2 \cdots v_k \) inductively by the following greedy strategy. Assume that we have defined the factors \( v_1, v_2, \ldots, v_{i-1} \) such that \( w = v_1 \cdots v_{i-1} w' \) and \( q_0 \overset{w'}{\rightarrow} q_{i-1} \). The factor \( v_i \) is defined as the label of the longest possible initial segment \( \pi_i \) of the path \( q_{i-1} \overset{v_i}{\rightarrow} q' \) such that either \( \pi_i \) contains a cycle over \( \text{alph}(v_i) \) or \( \pi_i = q_{i-1}, a, q_i \), where \( v_i = a \), thus \( v_i = 1 \). Such a factorization is well defined, and it is a cyclic factorization of \( w \).

Let \( p_i \), for \( i = 1, 2, \ldots, k \), be a state such that the path \( q_{i-1} \overset{v_i}{\rightarrow} q_i \) contains a cycle \( p_i \rightarrow p_i \) over the alphabet \( \text{alph}(v_i) \) if \( v_i \) is a cycle factor, and \( p_i = q_{i-1} \) if \( v_i \) is a letter factor. If \( p_1 \neq p_j \) with \( i < j \) such that \( v_i \) and \( v_j \) are cycle factors, then
factors, then we have a contradiction with the maximality of \( v_1 \) since \( q_{i-1} \xrightarrow{v_i} q_j \) contains a cycle \( p_i \to p_i \) from \( p_i \) to \( p_j \), over the alphabet \( \text{alph}(v_1, v_2, \ldots, v_j) \). Therefore, the factorization contains at most \( n \) cycle factors.

Note that \( v_1 \) is a letter factor only if the state \( p_i \), which is equal to \( q_{i-1} \) in such a case, has no reappearance in the path \( q_{i-1} \xrightarrow{v_i} q_j' \). This implies that there are at most \( n - 1 \) letter factors. Finally, if \( \text{alph}(v_i) = \text{alph}(w) \), then \( v_i = v_1 = w \) follows from the maximality of \( v_1 \).

We now define inductively cyclic factorizations of \( w_i \) such that the factorization of \( w_{i-1} \) is a refinement of the factorization of \( w_i \). Let \( w_i = v_{i,1}v_{i,2} \cdots v_{i,k} \) be a cyclic factorization of \( w_i \) defined, as described above, by some accepting path in the automaton \( A_i \mod 2 \). Factorizations \( w_{i-1} = v_{i-1,1}v_{i-1,2} \cdots v_{i-1,k_i} \) are defined as follows. Let \( w_{i-1} = v_{i,j_i}'v_{i,j_2}' \cdots v_{i,k_i}' \), where \( v_{i,j}' \) is the same for each \( j = 1, 2, \ldots, k_i \). Note that such a factorization exists, since we have that \( w_i < W_i \). Then \( v_{i-1,j_i}v_{i-1,j_2} \cdots v_{i-1,k_i} \) is defined as a concatenation of cyclic factorizations of \( v_{i,j}' \), for \( j = 1, 2, \ldots, k_i \), corresponding to an accepting path of \( w_{i-1} \) in \( A_i \mod 2 \). The cyclic factorization of the empty string is defined as empty. Note also that a letter factor of \( w_i \) either disappears in \( w_{i-1} \), or it is “factored” into a letter factor.

In order to measure the height of a tower, we introduce a weight function \( f \) of factors in a factorization \( v_1v_2 \cdots v_k \). First, let \( g(x) = \frac{n^x - 1}{n - 1} \). Note that \( g \) satisfies \( g(x + 1) = n \cdot g(x) + (n - 1) + 1 \). Now, let \( f(v_i) = 1 \) if \( v_i \) is a letter factor, and let \( f(v_i) = g(\text{alph}(v_i)) \) if \( v_i \) is a cycle factor. Note that, by definition, \( f(\varepsilon) = 0 \). The weight of the factorization \( v_1v_2 \cdots v_k \) is then defined by \( W(v_1v_2 \cdots v_k) = \sum_{v_i \in v} f(v_i) \). Let \( W_i = W(v_1v_2 \cdots v_k) \). We claim that \( W_{i-1} < W_i \) for each \( i = 2, 3, \ldots, r \). Let \( v_1v_2 \cdots v_k \) be the fragment of the cyclic factorization of \( w_{i-1} \) that emerged as the cyclic factorization of \( v_{i,j}' \). Then, if the factorization is not trivial, then, by the above analysis, \( W(v_1v_2 \cdots v_k) \leq n - 1 + n \cdot g(\text{alph}(v_{i,j})) - 1 < g(\text{alph}(v_{i,j})) = f(v_{i,j}) \). Similarly, we have \( f(v_{i,j}') < f(v_{i,j}) \) if \( |\text{alph}(v_{i,j}')| < |\text{alph}(v_{i,j})| \). Altogether, we have \( W_{i-1} < W_i \) as claimed, unless \( k_{i-1} = k_i \), the factor \( v_{i-1,j_i} \) is a letter factor if and only if \( v_{i,j_i} \) is a letter factor, and \( \text{alph}(v_{i-1,j_i}) = \text{alph}(v_{i,j_i}) \) for all \( j = 1, 2, \ldots, k_i \). Assume that such a situation takes place. We show that it leads to an infinite tower. Let \( L \) be the language of strings \( z_1z_2 \cdots z_k \) such that \( z_j = v_{i,j_i} \) if \( v_{i,j_i} \) is a letter factor, and \( z_j \in (\text{alph}(v_{i,j_i}))^* \) if \( v_{i,j_i} \) is a cycle factor. Since \( w_i \in L(A_i \mod 2) \) and \( w_{i-1} \in L(A_{i-1} \mod 2) \), the definition of a cycle factor implies that, for each \( z \in L \), there exist \( z' \in L(A_0) \cap L \) such that \( z \leq z' \) and \( z'' \in L(A_1) \cap L \) such that \( z \leq z'' \). The existence of an infinite tower follows. We have therefore proved that \( W_{i-1} < W_i \).

The proof is completed, since \( W_r \leq f(w_r) \leq g(m) \), \( W_1 \geq 0 \), and the bound in the claim is equal to \( g(m) + 1 \).

6. The lower bound on the height of towers

It was shown in [13] that the upper bound is tight for binary regular languages up to a linear factor. Namely, it was shown that for every odd positive integer \( n \), there exist two binary NFAs with \( n - 1 \) and \( n \) states having a tower of height \( n^2 - 4n + 5 \) and no infinite tower. The following theorem improves this result.

**Theorem 7.** For every even positive integer \( n \), there exists a binary NFA with \( n \) states and a binary DFA with \( n \) states having a tower of height \( n^2 - 4n + 5 \) and no infinite tower.

**Proof.** Let \( n \) be an even number and define the automata \( A_0 \) and \( A_1 \) with \( n \) states as depicted in Figure 3. The NFA \( A_0 = \{(1, 2, \ldots, n), (a, b), \delta_0, [1, 2, \ldots, n - 1], (n - 1)\} \) consists of an \( a \)-path through states \( 1, 2, \ldots, n - 1 \), of self-loops under \( b \) in all states \( 1, 2, \ldots, n - 2 \), and of a \( b \)-cycle from \( n - 1 \) to \( n \) and back to \( n - 1 \).

The DFA \( A_1 = \{(1, 2, \ldots, n), (a, b), \delta_1, [1, 2, 4, \ldots, n]\} \) consists of a \( b \)-path through states \( 1, 2, \ldots, n \) and of an \( a \)-transition from state \( n \) to state \( 1 \). All even states are accepting.
Consider the string \( w = (b^{p+1}a)^{p+2} b^k \). Note that \( A_0 \) accepts all prefixes of \( w \) ending with an even number of \( bs \), including those ending with \( a \). In particular, \( w \in L(A_0) \). On the other hand, the automaton \( A_1 \) accepts all prefixes of \( w \) ending with an odd number of \( bs \). Hence the maximum height of a tower between \( L(A_0) \) and \( L(A_1) \) is at least \( n(n - 2) + n + 1 = n^2 - n + 1 \).

To see that there is no infinite tower between \( L(A_0) \) and \( L(A_1) \), notice that any string in \( L(A_0) \) contains at most \( n - 2 \) occurrences of letter \( a \). As the languages are disjoint (they require a different parity of the \( b \)-tail), any infinite tower has to contain a string from \( L(A_1) \) of length more than \( n \cdot (n - 1) \). But any such string of \( L(A_1) \) contains at least \( n - 1 \) occurrences of letter \( a \), hence it cannot be embedded into any string of \( L(A_0) \). This means that there cannot be an infinite tower.

Assuming that the alphabet is fixed, it was shown in [13] that for every \( n \geq 1 \), there exist two NFAs with at most \( n \) states over a four-letter alphabet having a tower of height \( \Omega(n^2) \) and no infinite tower. However, since our original motivation comes from the XML databases, we are more interested in the case where the alphabet grows with the size of the automata.

**Theorem 8.** For every \( m \geq 0 \), there exist an NFA with \( m + 1 \) states and a DFA with two states over an alphabet of cardinality \( m + 1 \) having a tower of height \( 2^{m+1} \) and no infinite tower.

**Proof.** Let \( m \) be a non-negative integer. We define a pair of NFAs \( A_m \) and \( B_m \) over the alphabet \( \Sigma_m = \{b, a_1, a_2, \ldots, a_m\} \) with a tower of height \( 2^{m+1} \) between \( L(A_m) \) and \( L(B_m) \), and with no infinite tower.

The transition function \( \gamma_m \) of the NFA \( A_m = ((0, 1, 2, \ldots, m), \Sigma_m, \gamma_m, [0, 1, 2, \ldots, m], [0]) \) consists of the self-loop under \( b \) for all states but 0, self-loops under \( a_i \) for all states \( j > i \), and \( a_j \)-transitions from state \( i \) to state \( j < i \). Formally, \( \gamma_m(i, b) = \{b\} \) if \( i = 1, 2, \ldots, m \), \( \gamma_m(i, a_j) = \{i\} \) if \( m \geq i > j \geq 1 \), and \( \gamma_m(i, a_i) = \{0, 1, \ldots, i - 1\} \) if \( m \geq i > 1 \).

The NFAs \( A_1, A_2 \) and \( A_3 \) are shown in Figure 4. Note that \( A_m \) is an extension of \( A_{m-1} \), in particular, \( L(A_{m-1}) \subseteq L(A_m) \).

We set \( L(B_m) = \Sigma_m^* b \). The two-state NFA \( B_m = ((1, 2), \Sigma_m, \delta_m, \{1\}, \{2\}) \) consists of self-loops under all letters in state 1 and a \( b \)-transition from state 1 to the accepting state 2. The minimal DFA obtained from \( B_m \) has two states, which fulfills the statement of the theorem.

Define the string \( u_m \) inductively by \( u_0 = \varepsilon \) and \( u_k = u_{k-1} ba_k u_{k-1} \), for \( 0 \leq k \leq m \). Note that \( |u_k b| = 2^{k+1} - 1 \). It is not hard to see that every prefix of \( u_m b \) of odd length ends with \( b \), therefore it is an element of \( L(B_m) \). We now show, by induction on \( m \), that every prefix of \( u_m b \) of even length is accepted by \( A_m \). Indeed, the empty string is accepted by \( A_0 = (\{0\}, \{b\}, \gamma_0, \{0\}, \{0\}) \), an automaton with no transitions. Consider a prefix \( v \) of \( u_m b \) of even length. If \( |v| \leq 2^m - 1 \), then \( v \) is a prefix of \( u_{m-1} b \), hence, by the induction hypothesis, \( v \in L(A_{m-1}) \subseteq L(A_m) \). If \( |v| > 2^m - 1 \), then \( v = u_{m-1} ba_m v' \), where \( u_{m-1} b v' \) are over \( \Sigma_{m-1} \). By the induction hypothesis, \( v' \) is accepted in \( A_m \) from a state \( \ell \in \{0, 1, \ldots, m - 1\} \). The definition of \( A_m \) yields that there is a path

\[
m \xrightarrow{u_m b} m \xrightarrow{u_k} \ell \xrightarrow{} 0.
\]

Thus, we have proved that the prefixes of \( u_m b \) form a tower between \( L(A_m) \) and \( L(B_m) \) of height \( 2^{m+1} \).

It remains to show that there is no infinite tower. Again, we can use the techniques of [5, 23]. However, to give a brief idea, notice that there is no infinite tower between \( L(A_0) \) and \( L(B_0) \). Consider a tower between \( L(A_{m+1}) \) and \( L(B_{m+1}) \). If every string of the tower belonging to \( L(A_{m+1}) \) is accepted from an initial state different from \( m + 1 \), it is a tower between \( L(A_m) \) and \( L(B_m) \), hence it is finite by the induction hypothesis. Thus, if there is an infinite tower,
there is also an infinite tower where all the strings of \( L(A_{m+1}) \) are accepted from state \( m + 1 \). Every such string is of the form \( \sum_m a_{m+1} v \), where \( v \) is accepted from an initial state different from \( m + 1 \). Cutting off the prefixes from \( \sum_m a_{m+1} \) results in an infinite tower between \( L(A_m) \) and \( L(B_m) \), which is a contradiction. 

It is worth mentioning that the languages \( L(A_m) \) are piecewise testable, hence they are the separators. The easiest way to see this is to transform \( A_m \) to its minimal DFA and to use Trahtman’s algorithm [33].

In the construction of the tower in the proof of Theorem 8, the simplicity of the automaton \( B_m \) is conspicuous. Introducing the structure present in the automaton \( A_m \) also into \( B_m \) can further quadratically increase the height of the tower. However, the nature of the exponential bound implies that the blow-up is only a constant in the exponent.

**Theorem 9.** For every \( m \geq 1 \), there exist two NFAs with \( m + 1 \) states over an alphabet of cardinality \( 2m \) having a tower of height \( \Omega(2^m) \) and no infinite tower.

**Proof.** For every non-negative integer \( m \geq 1 \), we define a pair of NFAs \( A'_m \) and \( B'_m \) with \( m + 1 \) states over the alphabet \( \Sigma_m = \sum_m \cup \{c_1, c_2, \ldots, c_{m-1}\} \), where \( \sum_m = \{a, b, c_1, c_2, \ldots, c_m\} \) as in Theorem 8 with a tower of height \( 2^{m^2} - 1 \) + 2 between \( L(A'_m) \) and \( L(B'_m) \), and such that there is no infinite tower. Let \( Q_m = \{0, 1, \ldots, m\} \).

The automaton \( A'_m = (Q_m, \Sigma_m, \gamma'_m, Q_m, \{0\}) \) is obtained from the automaton \( A_m \) of the proof of Theorem 8 by adding transitions from state 0 to each state of \( Q_m \) under the letters \( c_1, c_2, \ldots, c_{m-1} \).

The automaton \( B'_m = (Q_m, \Sigma_m, \delta'_m, Q_m \setminus \{0\}, \{0\}) \) contains self-loops in each state \( k \in \{1, 2, \ldots, m\} \) under the alphabet \( \Sigma_m \cup \{c_1, c_2, \ldots, c_{k-2}\} \), transitions under \( c_{k-1} \) from state \( k \) to state \( i \) for each \( k = 2, 3, \ldots, m \) and \( 1 \leq i < k \), and a \( b \)-transition from state 1 to state 0.

The automata \( A'_m \) and \( B'_m \) are shown in Figure 5 and the automata \( A'_1 \) and \( B'_1 \) in Figure 6.

We now define a string \( \omega_m \) using the strings \( u_i \) from the proof of Theorem 8 as follows. For \( k = 1, 2, \ldots, m-1 \), let \( \omega_{m,0} = u_m \), \( \omega_{m,k} = \omega_{m,k-1} c_k \omega_{m,k-1} \), and \( \omega_m = \omega_{m,m-1} b \). The string \( \omega_m \) consists of \( 2^{m-1} \) occurrences of the string \( u_m \) separated by some letters \( c_i \). We prove by induction on \( k = 0, 1, \ldots, m-1 \) that every prefix \( v \) of \( \omega_{m,k} \) ending with \( b \) is accepted by \( B'_m \) from an initial state \( \ell \), \( \ell \leq k + 1 \), and every prefix of \( \omega_{m,k} \) ending with some \( a_i \) is accepted by \( A'_m \). This is true for \( \omega_{m,0} \) by Theorem 9, since \( A'_m \), restricted to the alphabet \( \Sigma_m \), yields \( A_m \), and \( B'_m \) accepts all strings over \( \Sigma_m \) ending with \( b \) from state 1. Let \( k \geq 1 \) and consider the string \( \omega_{m,k-1} \). The claim holds for prefixes of \( \omega_{m,k-1} \) by induction.

Consider a prefix \( v = \omega_{m,k-1} c_k v' \) ending with \( a_i \). By the induction hypothesis, \( A'_m \) accepts the string \( \omega_{m,k-1} \) from some state \( i_1 \), and it accepts \( v' \) from some initial state \( i_2 \neq 0 \), since \( v' \) does not begin with a \( c_i \). Therefore, \( A'_m \) accepts \( v \) by the path \( i_1 \stackrel{\omega_{m,k-1}}{\rightarrow} 0 \rightarrow i_2 \rightarrow v' \).
Let now \( v = \omega_{m,k-1} \omega' \) end with \( b \). By the induction hypothesis, \( \omega' \) is accepted by \( B'_m \) from a state \( \ell_{\omega'} \leq k \), since \( \omega' \) is a prefix of \( \omega_{m,k-1} \). Moreover, \( \omega_{m,k-1} \) contains no letter \( c_i \), for \( i > k-1 \). Thus, the automaton \( B'_m \) accepts \( v \) by the path \( k + 1 \overset{\omega_{m,k-1}}{\rightarrow} k + 1 \overset{\omega'}{\rightarrow} \ell_{\omega'} \overset{v}{\rightarrow} 0 \). This proves the claim.

Since \( A'_m \) accepts \( \varepsilon \) and \( B'_m \) accepts \( \omega_m \), we have a tower of height \( 2^{m-1}(2^{m+1} - 2) + 2 \), where \( 2^{m+1} - 2 \) is the length of \( u_m \).

To prove that there is no infinite tower, we can use a similar argument as in the proof of Theorem 8, where \( B'_m \) plays the role of \( A_m \) and \( c_i \) the role of \( a_i \), and the fact that there is no infinite tower between \( A_m \) and \( B_m \) over the alphabet \( \Sigma_m \).

\[ \square \]

6.1. The lower bound for DFAs

The exponential lower bounds presented above are based on NFAs. It is, however, an interesting question whether they can also be achieved for DFAs. We discuss this problem below.

**Theorem 10.** For every \( n \geq 0 \), there exist two DFAs with at most \( n + 1 \) states over an alphabet of cardinality \( \frac{2^{n+1}}{2} + 1 \) having a tower of height \( 2^n \) and no infinite tower.

**Proof.** The main idea of the construction is to “determinize” the automata of the proof of Theorem 8. Thus, for every non-negative integer \( n \), we define a pair of deterministic automata \( A_n \) and \( B_n \) with \( n + 1 \) and two states, respectively, over the alphabet \( \Sigma_n = \{b\} \cup \{a_i \mid i = 1, 2, \ldots, n; j = 0, 1, \ldots, i-1\} \) with a tower of height \( 2^n \) between \( L(A_n) \) and \( L(B_n) \), and with no infinite tower. The two-state DFA \( B_n = (\{1, 2\}, \Sigma_n, \gamma_n, 1, \{2\}) \) accepts all strings over \( \Sigma_n \) ending with \( b \) and is shown in Figure 7 (right).

The “determinization” idea of the construction of the DFA \( A_n = (\{0, 1, \ldots, n\}, \Sigma_n, \delta_n, n, \{0\}) \) is to use the automaton \( A_n \) from the proof of Theorem 8 and to eliminate nondeterminism by relabeling every transition \( i \overset{a_i}{\rightarrow} j \) with a new unique letter \( i \overset{a_i}{\rightarrow} j \). Then the tower of Theorem 8 is modified by relabeling the corresponding letters. However, to
that is, in $w$ and we have

By induction, we verify that this definition fits the above definition of $w$.

Every string $w_n(2i)$ is obtained from the string $w_n(2i + 1)$ by removing the last letter, which is $b$, and every string $w_n(2i - 1)$ is obtained from the string $w_n(2i)$ by replacing some $a_k$ with its suffix, that is, with $a_{k,j}$, where $j ≤ i$, or with the empty string, see Figure 9 for illustration (as usual for DFAs, all undefined transitions go to a new sink state that is not depicted for simplicity).

More explicitly, the tower is defined recursively using the towers for $A_k$ and $B_k$, where $1 ≤ k < n$. For any $k ≥ 1$, we define $w_k(0) = a_{k,0}a_{k,0}$ and $w_k(1) = a_{k,0}b$. For $i ≥ 2$, let

$$w_k(i) = a_{k,\log i} w_{\log i - 1} b w_{\log i} \left( i - 2^{\log i} \right).$$

By induction, we verify that this definition fits the above definition of $w_n$. Since $\log (2^n - 1) = n - 1$, we obtain that $w_n(2^n - 1) = a_{n-1} a_{n-2} b w_{n-1}(2^{n-1} - 1) = a_{n-1} a_{n-2} b a_{n-2} a_{n-2} b = a_{n-1} a_{n-2} b$. By (1), the relationship between $w_n(i)$ and $w_n(i + 1)$ is in most cases directly induced by the relationship between $w_{\log i}(i - 2^{\log i})$ and $w_{\log i}(i - 2^{\log i} + 1)$. A special case is when $i$ is of the form $2^\ell - 1$ for some $\ell > 1$. Then $\ell - 1 = \lceil \log i \rceil + \lceil \log (i + 1) \rceil = \ell$ and we have

$$w_n(2^\ell - 1) = a_{n,\ell-1} a_{\ell-2} b w_{\ell-1}(2^{\ell-1} - 1) = a_{n,\ell-1} a_{\ell-2} b a_{\ell-2} a_{\ell-2} b = a_{n,\ell-1} a_{\ell-1} b,$$

$$w_n(2^\ell) = a_{n,\ell} a_{\ell-1} b w_{\ell}(0),$$

that is, $w_n(i + 1) = a_{n,\ell} w_n(i) a_{\ell,0}$.

We now prove that the sequence is the required tower. If $n = 1$, the tower is $a_{1,0} a_{1,0} b$. Let $n > 1$. The definition implies that $w_n(i)$ is in $L(B_n)$ (that is, it ends with $b$) if and only if $i$ is odd. Consider $w_n(i)$ with even $i ≥ 2$. The path in $A_n$ defined by $w_n(i)$ can be by (1) decomposed as

$$n \xrightarrow{a_{n,\log i}} \lceil \log i \rceil \xrightarrow{a_{n,\log i - 1} a_{\log i - 1} b} \lceil \log i \rceil \xrightarrow{w_{\log i}(i - 2^{\log i})} 0.$$
For the second part, note that both the alphabet of \(a_{m, [\log i - 1]} \) and the alphabet \([b] \cup \{a_{m,m'} | m \leq [\log i - 1], m' < m\}\) of \(u_{[\log i - 1]}b\) are contained in the alphabet of self-loops of state \([\log i]\). The last part of the path follows by induction, since \([\log i] < n, i - 2[\log i] \leq 2[\log i] - 1, \) and \(i - 2[\log i]\) is even.

Finally, we observe that \(w_x(i) \leq w_x(i + 1)\). This follows by induction from (1) if \([\log(i + 1)] = [\log i]\), and from (2) if \(i = 2^t - 1\).

We now prove that the “determinization” idea of the previous theorem can be generalized. However, compared to the proof of Theorem [11], the general procedure suffers from the increase of states. The reason why we do not need to increase the number of states in the proof of Theorem [11] is that there is an order in which the transitions/states are used/visited, and that the nondeterministic transitions are acyclic.

**Theorem 11.** For every two NFAs \(A\) and \(B\) with at most \(n\) states and \(m\) input letters, there exist two DFAs \(A'\) and \(B'\) with \(O(n^3)\) states and \(O(m + n)\) input letters such that there is a tower of height \(r\) between \(A\) and \(B\) if and only if there is a tower of height \(r\) between \(A'\) and \(B'\). In particular, there is an infinite tower between \(A\) and \(B\) if and only if there is an infinite tower between \(A'\) and \(B'\).

**Proof.** Let \(A\) and \(B\) be two NFAs with at most \(n\) states over an alphabet \(\Sigma\) of cardinality \(m\). Without loss of generality, we may assume that the automata each have a single initial state. Let \(Q_A\) and \(Q_B\) denote their respective sets of states. We modify the automata \(A\) and \(B\) to obtain the DFAs \(A'\) and \(B'\) as follows. Let \(Q_{A'} = Q_A \cup \{\sigma_{s,h} | s, h \in Q_A\}\) and \(Q_{B'} = Q_B \cup \{\sigma_{s,t} | s, t \in Q_B\}\), where \(\sigma_{s,t}\) are new states. We introduce a new letter \(y_i\) for every state \(t \in Q_{A} \cup Q_{B}\). It results in \(O(n^2)\) states and \(O(m + n)\) letters. The transition function is defined as follows.

In both automata, each transition \(s \xrightarrow{a} t\) is replaced with two transitions \(s \xrightarrow{\sigma_{s,t}} \sigma_{s,t} \xrightarrow{a} t\). Moreover, self-loops in all new states are added over all new letters. Note that all transitions are deterministic in \(A'\) and \(B'\).

We now prove that if there is a tower of height \(r\) between \(A\) and \(B\), then there is a tower of height \(r\) between \(A'\) and \(B'\). Let \((w_i)_{i=1}^{r+1}\) be a tower between \(A\) and \(B\). Let

\[w_i = x_{i,1}x_{i,2}\cdots x_{i,n},\]

where \(n = |w_i|\) and \(x_{i,j}\) is either a letter or the empty string such that \(x_{i,j} \leq x_{i+1,j}\) for each \(i = 1, 2, \ldots, r - 1\) and \(j = 1, 2, \ldots, n\). For every \(w_i\), we fix an accepting path \(\pi_i\) in the corresponding automaton. Let \(p_{i,j}\) be the letter \(y_j\) where 

\[s \xrightarrow{\sigma_{s,t}} \sigma_{s,t} \xrightarrow{t} t\]

Moreover, self-loops in all new states are added over all new letters. Note that all transitions are deterministic in \(A'\) and \(B'\).

Let \(w_i' = \alpha_{i,1}\alpha_{i,2}\cdots \alpha_{i,n}\), where \(\alpha_{i,j} = p_{i,j}p_{i-1,1}\cdots p_{1,0}\) if \(x_{i,j} = a\), and empty otherwise. It is straightforward to verify that \((w_i')_{i=1}^{r+1}\) is a tower of height \(r\) between \(A'\) and \(B'\).

Let now \((w_i')_{i=1}^{r+1}\) be a tower between \(A'\) and \(B'\). We show that \((p(w_i'))_{i=1}^{r+1}\) is a tower between \(A\) and \(B\), where \(p\) is a projection erasing all new letters. Obviously, we have \(p(w_i') \leq p(w_{i+1}')\). We now show that if a string \(w\) is accepted by \(A'\), then \(p(w')\) is accepted by \(A\). Let \(\pi\) be the path accepting \(w\), and let \(t_1', t_2', \ldots, t_k'\) denote the sequence of all transitions of \(\pi\) labeled with letters from \(\Sigma\) in the order they appear in \(\pi\). By construction, \(t_i'\) is of the form \(\sigma_{s,t} \xrightarrow{a} t\). Let \(t_i\) be \(s \xrightarrow{\sigma_{s,t}} t\). We claim that \(t_1, t_2, \ldots, t_k\) is an accepting path of \(p(w')\) in \(A\). Indeed, the transitions \(t_i\) are valid transitions in \(A\) by the construction of \(A'\). Let \(t_i', i < k\), end in \(s\). Then the successive state in \(\pi\) is \(\sigma_{s,t}\), for some \(t \in Q_A\), and the transition \(t_i'\) must be \(\sigma_{s,t} \xrightarrow{a} t\), for some \(a \in \Sigma\), since all \(\Sigma\)-transitions from \(\sigma_{s,t}\) end in \(t\), and all other transitions are self-loops. This proves that the transitions \(t_1, t_2, \ldots, t_k\) form a path in \(A\), which is accepting, since \(\pi\) is accepting and \(A\) has the same initial state and accepting states as \(A'\). Analogously for \(B'\) and \(B\). The proof is completed by [8, Lemma 6], which shows that there is an infinite tower if and only if there is a tower of arbitrary height.

A similar construction yields the following variant of the previous theorem.

**Theorem 11.** For every two NFAs \(A\) and \(B\) with at most \(n\) states and \(m\) input letters, there exist two DFAs \(A'\) and \(B'\) with \(O(mn)\) states and \(O(mn)\) input letters such that there is a tower of height \(r\) between \(A\) and \(B\) if and only if there is a tower of height \(r\) between \(A'\) and \(B'\). In particular, there is an infinite tower between \(A\) and \(B\) if and only if there is an infinite tower between \(A'\) and \(B'\).
Proof. Let \( Q_A = Q_1 \cup \{ \sigma_{a,t} \mid a \in \Sigma, t \in Q_A \} \) and \( Q_B = Q_2 \cup \{ \sigma_{a,t} \mid a \in \Sigma, t \in Q_A \} \), where \( \sigma_{a,t} \) are new states. New letters are \( a_i \) for every state \( t \in Q_A \cup Q_B \) and every letter \( a \in \Sigma \). We have \( O(mn) \) states and letters. Each transition \( s \overset{a}{\rightarrow} t \), in both automata, is replaced with two transitions \( s \overset{a_i}{\rightarrow} \sigma_{a,t} \) and \( \sigma_{a,t} \overset{a}{\rightarrow} t \). Self-loops in all new states are added over all new letters.

The rest of the proof is analogous to the proof of Theorem 11. We just need to slightly modify the argument that \( \tau_1, \tau_2, \ldots, \tau_k \) is an accepting path of \( p(w') \) in \( A \). Let \( \sigma_{a,t} \overset{a}{\rightarrow} t \) and \( \sigma_{b,q} \overset{b}{\rightarrow} q \) be two successive transitions with labels from \( \Sigma \) in \( \pi' \). Then the transition \( \sigma_{a,t} \overset{a}{\rightarrow} t \) is in \( \pi' \) necessarily followed by a transition \( t \overset{b}{\rightarrow} \sigma_{b,q} \), which shows that \( t \overset{b}{\rightarrow} q \) is a valid transition in \( A \) starting in the final state of the previous transition. This was to be shown. \( \square \)

7. Towers of prefixes

The definition of towers can be generalized from the subsequence relation to any relation on strings. However, we are particularly interested in towers of prefixes. The reason is that the lower bounds on the height of finite towers for NFAs have been obtained by towers of prefixes, that is, every string \( w_i \) of the tower is a prefix of the string \( w_{i+1} \). Even though we are not aware of any tower for \( n \)-state NFAs that would be higher then the bound on towers of prefixes for \( n \)-state NFAs (see below), the analysis of this section indicates that such towers exist. On the other hand, if the automata are deterministic, we show that the difference in the height between towers and towers of prefixes is exponential, cf. Theorem 10 and the results below.

It is obvious that any tower of prefixes is also a tower (of subsequences). Hence, the non-existence of an infinite tower implies the non-existence of an infinite tower of prefixes. However, the existence of an infinite tower does not imply the existence of an infinite tower of prefixes. This can be easily seen by considering the languages \( L_1 = a(ab)_n^* \) and \( L_2 = b(ab)_n^* \). Indeed, there is no infinite tower of prefixes, since every string of \( L_1 \) begins with \( a \) and thus cannot be a prefix of a string of \( L_2 \), and vice versa. But there is an infinite tower, namely, \( a, bab, ababa, \ldots. \)

In [8], it was shown that if there exist towers of arbitrary height, then there exists an infinite tower. More precisely, this property was shown to hold for any relation that is a well quasi order (WQO), in particular for the subsequence relation. Since the prefix relation is not a WQO, the property does not hold for it in general. However, it still holds for regular languages.

Lemma 12. Let \( K \) and \( L \) be regular languages. If there are towers of prefixes of arbitrary height between \( K \) and \( L \), then there is an infinite tower of prefixes between them.

Proof. Assume that the languages are given by minimal DFAs \( A = (Q_A, \Sigma, \delta_A, q_A, F_A) \) and \( B = (Q_B, \Sigma, \delta_B, q_B, F_B) \) and that there is no infinite tower of prefixes between \( L(A) \) and \( L(B) \). In particular, the languages are disjoint. Consider the product automaton \( A \times B = (Q_A \times Q_B, \Sigma, \delta, (q_A, q_B), \emptyset) \). Let \( (w_i')_{i=1}^\infty \) be a tower of prefixes between \( L(A) \) and \( L(B) \). It defines a corresponding sequence of states \( (\delta(q_A, q_B), w_i)'_{i=1}^\infty \). Assume that \( r > |Q_A \times Q_B| \). Then there exists a state, \( (p, q) \), that appears at least twice in the sequence. Let \( v_1, v_2 \in \{ w_1, w_2, \ldots, w_r \} \) be two different strings (of the same language \( L(A) \) or \( L(B) \)) such that \( \delta(q_A, q_B, v_j) = (p, q) \), for \( j = 1, 2 \), and \( v_1 \neq v_2 \). Because \( v_1 \) and \( v_2 \) belong to the same language, there must exist \( v'_1 \in \{ w_1, w_2, \ldots, w_r \} \) from the other language such that \( v_1 \neq v'_1 \). Thus, \( v'_1 = v_1 x \) and \( v_2 = v_1 y \), for some nonempty strings \( x \) and \( y \). We can now go through the cycle and define an infinite tower of prefixes \( v_1, v_1 x, v_1 xy, v_1 xy x, \ldots. \) which is a contradiction. Thus, the bound on \( r \) is at most the number of states of the product automaton of the minimal DFAs.

We now show that Lemma 12 does not hold for non-regular languages.

Example 1. Let \( K = \{a, b\}^*a \) and \( L = \{a^m(ba^n)^*b \mid m \geq n \geq 0 \} \) be two languages. Note that \( K \) is regular and \( L \) is non-regular context-free. The languages are disjoint, since the strings of \( K \) end with \( a \) and the strings of \( L \) with \( b \).

For any \( k \geq 1 \), the strings \( w_{2i+1} = a^k(ba)^i \in K \) and \( w_{2i+1} = a^k(ba)^i b \in L \), for \( i = 0, 1, \ldots, k-1 \), form a tower of prefixes between \( K \) and \( L \) of height \( 2k \).

On the other hand, let \( w_1, w_2, \ldots. \) be a tower of prefixes between the languages \( K \) and \( L \). Without loss of generality, we may assume that \( w_1 \) belongs to \( L \). Then \( a^k b \) is a prefix of \( w_1 \), for some \( k \geq 1 \). It is not hard to see that \( |w_{2i+1}| < |w_{2i+2}| \) holds for any \( w_1 \leq w_{2i+1} \leq w_{2i+2} \) with \( w_1, w_{2i+2} \) in \( L \) and \( w_{2i+1} \) in \( K \). As any string of \( L \) with a prefix \( a^k b \) can have at most \( k \) occurrences of letter \( b \), the tower cannot be infinite.
7.1. The upper and lower bounds

We first investigate towers of prefixes for regular languages represented by DFAs.

**Theorem 13.** Let $A$ and $B$ be two DFAs with $m$ and $n$ states that have no infinite tower of prefixes. Then the height of a tower of prefixes between $A$ and $B$ is at most $\frac{m}{n}$, and the bound is tight.

**Proof.** Let $A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ and $B = (Q_B, \Sigma, \delta_B, q_B, F_B)$, and let $X = F_A \times (Q_B \setminus F_B)$ and $Y = (Q_A \setminus F_A) \times F_B$.

We first show that $\min(|X|, |Y|) \leq \frac{m}{n}$, where the equality holds only if $|X| = |Y|$. Let $0 \leq \alpha, \beta \leq 1$ be such that $|F_A| = \alpha|Q_A|$ and $|F_B| = \beta|Q_B|$. Then $|X| = \alpha(1 - \beta)mn$ and $|Y| = \beta(1 - \alpha)mn$. The well-known inequality between the arithmetic and geometric mean yields $\frac{1}{2}[\alpha(1 - \beta) + \beta(1 - \alpha)] = \frac{1}{2} \geq \sqrt{\alpha(1 - \alpha)\beta(1 - \beta)}$. This implies the claim.

If there is no infinite tower between $A$ and $B$, any tower of prefixes $(w_i)_{i=1}^k$ has to alternate between the states of $X$ and $Y$ in the product automaton $A \times B$, as in the proof of Lemma 12. If $r > \frac{m}{n}$, then the claim above implies that at least one state is repeated, and we obtain an infinite tower of prefixes, which is a contradiction.

To show that the bound is tight, we consider the automata $A_m$ and $B_n$ from the proof of Theorem 8 with a tower of prefixes of height $2^{m+1}$. The minimal DFA for $L(A_m)$ has two states. We claim that the minimal DFA for $L(A_m)$ has $2^{m+1}$ states. Indeed, consider the DFA obtained from $A_m$ by the standard subset construction. Let $Z$ and $Z'$ be two different subsets of states, and let $i \in Z \setminus Z'$. Then $a_i$ is accepted from $Z$, but not from $Z'$. It remains to show that all subsets are reachable from the initial state $I = \{0, 1, \ldots, m\}$. Recall that the $a_i$-transition in $A_m$ leads from $k$ to $j$ for all $j < k$, that there is a self-loop in $j$ under $a_i$ if $j > k$, and that the $a_i$-transition is not defined in $j$ if $j < k$. Therefore, if $k \in Z'$, the $a_i$-transition leads from $Z'$ to $Z''$, where $\{0, 1, \ldots, k - 1\} \subseteq Z'$ and, for $j > k$, $j \in Z''$.

Let $L = I \setminus \{i_1, i_2, \ldots, i_k\}$, where $i_1 > i_2 > \cdots > i_k$. It is straightforward to verify that the string $a_{i_1}a_{i_2}\cdots a_{i_k}$ leads from $I$ to $Z$.

Compared to towers (of subsequences), Theorem 10 shows that there exist towers (of subsequences) of exponential height with respect to the number of states of the input DFAs, while Theorem 13 gives a quadratic bound on the height of towers of prefixes with respect to the number of states of the input DFAs. This shows an exponential difference between the height of towers (of subsequences) and the height of towers of prefixes.

What is the situation for NFAs? An immediate consequence of the NFA-to-DFA transformation and Theorem 13 gives the following tight bound.

**Corollary 14.** Given two NFAs with at most $m$ and $n$ states and with no infinite tower of prefixes. The height of a tower of prefixes is at most $2^{m+n-1}$. Moreover, the lower bound is at least $2^{m+n-2}$ for infinitely many pairs $(m, n)$.

**Proof.** The automata $A_{m-1}$ and $B_{n-1}$ in Theorem 8 have $m$ and $2$ states, respectively, and the tower is a tower of prefixes of height $2^m$. Unlike the bounds on towers (of subsequences), which depend on both the number of states and the size of the alphabet, the bounds on towers of prefixes depend only on the number of states of the alphabet. A natural question is whether there are any requirements on the size of the alphabet in case of automata with exponentially high towers of prefixes. The following corollary shows that the alphabet can be binary and the tower is still more than polynomial in the number of states.

**Corollary 15.** There exist infinitely many pairs $(n_1, n_2)$ of integers for which there are binary NFAs with $n_1$ and $n_2$ states with no infinite tower of prefixes and with a tower of prefixes of a superpolynomial height with respect to $n_1 + n_2$.

**Proof.** The property of being a tower of prefixes is preserved if the alphabet is encoded in binary. The binary code of each letter has length at most $\log_2 m$ for an alphabet of cardinality $m$. Therefore, every transition under an original letter can be replaced by a path with at most $\log_2 m$ new states.

Consider again the automata $A_m$ and $B_n$ from Theorem 8. Every automaton $A_m$ has $m + 1$ states and $m(m + 1)$ transitions, and every automaton $B_n$ has two states and $n + 2$ transitions. Encoding every letter in binary results in automata with $n_1 = O(m^2 \log m)$ and $n_2 = O(m \log m)$ states, respectively, and a tower of prefixes of height $2^{m+1} = 2^{\Omega(m^2 \log m)}$.
7.2. The complexity analysis

In the rest of the paper, we study the complexity of the problem whether there exists an infinite tower of prefixes between two (nondeterministic) finite automata accepting disjoint languages.

Let $A = (Q_A, \Sigma, \delta_A, q^A_0)$ and $B = (Q_B, \Sigma, \delta_B, q^B_0)$ be two NFAs. We say that $(S, \sigma, \sigma_1, \sigma_2, \tau, \tau_1, \tau_2)$ is a pattern of the automata $A$ and $B$ if $S$ is a nontrivial (it has at least one edge) strongly connected component of the product automaton $A \times B$, and $\sigma, \sigma_1, \sigma_2, \tau, \tau_1, \tau_2$ are states of the product automaton such that

- $\sigma_1 \in F_A \times Q_B$ and $\tau_1 \in Q_A \times F_B$,
- $\sigma, \sigma_1, \tau, \tau_2$ belong to $S$,
- states $\sigma_1$ and $\sigma_2$ are reachable from the state $\sigma$ under a common string,
- states $\tau_1$ and $\tau_2$ are reachable from the state $\tau$ under a common string, and
- the strongly connected component $S$ is reachable from the initial state.

The definition is illustrated in Figure 10.

The following provides a characterization for the existence of an infinite tower of prefixes.

**Theorem 16.** Let $A$ and $B$ be two NFAs such that $L(A)$ and $L(B)$ are disjoint. Then there is a pattern of the automata $A$ and $B$ if and only if there is an infinite tower of prefixes between $A$ and $B$.

**Proof.** First, note that the states of $F_A \times F_B$ are not reachable from the initial state $q_0 = (q^A_0, q^B_0)$, because the languages are disjoint. Assume that $(S, \sigma, \sigma_1, \sigma_2, \tau, \tau_1, \tau_2)$ is a pattern of the automata $A$ and $B$. Let $u$ denote the shortest string under which $\sigma$ is reachable from the initial state $(q^A_0, q^B_0)$. Let $x$ (resp.) be a string under which both $\sigma_1$ and $\sigma_2$ are reachable from $\sigma$ (resp.). Let $u_1$ denote the shortest string under which $\tau$ is reachable from $\sigma_2$, and $u_2$ denote the shortest string under which $\sigma$ is reachable from $\tau_2$. Together, we have an infinite tower of prefixes $u(xu_1yu_2)^\infty(x + xu_1y)$.

To prove the other direction, assume that there exists an infinite tower of prefixes $(w_i)_{i=1}^\infty$ between the languages $L(A)$ and $L(B)$. Consider the automaton det$(A \times B)$, the determination of $A \times B$ by the standard subset construction. A sufficiently long element of the tower defines a path $q_0 \xrightarrow{u} x \xrightarrow{z_1} y \xrightarrow{z_2} X$ in the automaton det$(A \times B)$, such that $Y$ contains a state $(f_1, f_2) \in F_A \times (Q_B \setminus F_B)$ and $X$ contains a state $(q_2, f_2) \in (Q_A \setminus F_A) \times F_B$.

Since $Y = \delta_{A \times B}(X, z_2)$ and $X = \delta_{A \times B}(Y, z_1)$, for every state of $X$, there exists an incoming path from an element of $Y$ labeled by $z_2$. Similarly, for every state of $Y$, there exists an incoming path from an element of $X$ labeled by $z_1$. Thus, there are infinitely many paths from $X$ to $X$ labeled with $(z_1z_2)^\infty$ ending in state $(q_2, f_2)$. Therefore, there exists a state $(s_1, t_1) \in X$ and integers $k_1$ and $\ell_1$ such that $(s_1, t_1) \xrightarrow{(z_1z_2)^{k_1}} (s_1, t_1) \xrightarrow{(z_1z_2)^{\ell_1}} (q_2, f_2)$. Similarly, there exists a state $(s_2, t_2) \in X$ and integers $k_2$ and $\ell_2$ such that $(s_2, t_2) \xrightarrow{(z_1z_2)^{k_2}} (s_2, t_2) \xrightarrow{(z_1z_2)^{2\ell_2}} (f_1, q_1)$.

As $q_0 \xrightarrow{u} (s_1, t_1)$ and $q_0 \xrightarrow{u} (s_2, t_2)$, state $(s_2, t_1)$ belongs to $X$. Moreover, $(s_2, t_1) \xrightarrow{(z_1z_2)^{k_1+1}} (s_2, t_1)$ forms a cycle. Thus, state $(s_2, t_1)$ appears in a nontrivial strongly connected component of the automaton $A \times B$. From the above we
also obtain \((s_2, t_1) \xrightarrow{(xyz)^{r_1}} (s_1, f_2)\), for an \(s_3\) in \(Q_A\), which exists because \(s_2 \xrightarrow{(xyz)^{r_2}} s_2\) is a cycle in \(A\). Similarly, we obtain \((s_2, t_1) \xrightarrow{(xyz)^{r_3}z_2} (f_1, t_3)\), for a \(t_3\) in \(Q_B\). Notice that \((f_1, t_3) \in F_A \times (Q_B \setminus F_B)\) and \((s_3, f_2) \in (Q_A \setminus F_A) \times F_B\).

Thus, we have a pattern with \(\sigma = \tau = (s_2, t_1)\), \(\sigma_1 = (f_1, t_3)\), \(\tau_1 = (s_3, f_2)\), and with \(\sigma_2\) and \(\tau_2\) being states of the cycle \(\sigma \xrightarrow{(xyz)^{r_1}z_2} \tau \xrightarrow{(xyz)^{r_2}z_1} \sigma\), where \(x = (xyz)^{r_1}z_2y\) and \(y = (xyz)^{r_2}\) have been obtained above, and \(u_1\) and \(u_2\) can be chosen as \(u_1 = z_2y(xy)^{r_1}z_2^{k_2-1}\) and \(u_2 = (xy)^{r_2}z_2^{k_2-1}\).

We now use the previous result to study the complexity of the problem of the existence of an infinite tower of prefixes. The problem asks whether, given two (nondeterministic) finite automata accepting disjoint languages, there exists an infinite tower of prefixes between their languages.

**Theorem 17.** The problem of the existence of an infinite tower of prefixes is NL-complete.

**Proof.** The problem is in NL, since it is sufficient to guess the six states \(\sigma, \sigma_1, \sigma_2, \tau, \tau_1, \tau_2\) of a pattern and then it takes several reachability tests to verify the guess, see Algorithm 2. This can be done in nondeterministic logarithmic space [30].

**Algorithm 2** Checking the existence of a pattern of automata \(A\) and \(B\) (symbol \(\rightsquigarrow\) stands for reachability)

1. Guess six states \(\sigma, \sigma_1, \sigma_2, \tau, \tau_1, \tau_2\) such that \(\sigma_1 \in F_A \times Q_B\) and \(\tau_1 \in Q_A \times F_B\);
2. Check \(\sigma \rightsquigarrow \tau \rightsquigarrow \tau_1 \rightsquigarrow \sigma;\) \(\tau_2 \rightsquigarrow \tau;\) \(\sigma_2 \rightsquigarrow \sigma\); \(\sigma_1 \rightsquigarrow \sigma_2\);
3. Check reachability of \(\sigma\) from the initial state of \(A \times B\);
4. \(k_1 := \sigma;\) \(k_2 := \sigma;\)
5. repeat guess \(a \in \Sigma;\)
6. \(k_1 := \delta(k_1, a);\)
7. \(k_2 := \delta(k_2, a);\)
8. until \(k_1 = \sigma_1\) and \(k_2 = \sigma_2;\)
9. \(k_1 := \tau;\) \(k_2 := \tau;\)
10. repeat guess \(a \in \Sigma;\)
11. \(k_1 := \delta(k_1, a);\)
12. \(k_2 := \delta(k_2, a);\)
13. until \(k_1 = \tau_1\) and \(k_2 = \tau_2;\)
14. return ‘yes’;

To prove NL-hardness we reduce the reachability problem. The reachability problem asks, given a directed graph \(G\) and vertices \(s\) and \(t\), whether \(t\) is reachable from \(s\). The problem is NL-complete [24, Theorem 16.2].

Let \(G = (V, E, s, t)\) be an instance of the reachability problem. We construct the automaton \(A = (V, \Sigma, \delta_A, q_0, (q_0))\), where \(\delta_A\) is defined as the relation \(E\) where every transition is given a unique label. In addition, there are two more transitions \(\delta_A(q_0, a) = s\) and \(\delta_A(t, b) = q_0\), for some fresh letters \(a\) and \(b\). Thus, \(\Sigma = \{a, b\} \cup \{|e| \mid 1 \leq i \leq |E|\}\). The automaton \(B\) is depicted in Figure 11. It is not hard to see that there exists an infinite tower of prefixes if and only if \(t\) is reachable from \(s\).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{automaton.png}
\caption{Automaton \(A\) and \(B\).}
\end{figure}
Corollary 18. The problem of the existence of an infinite tower of prefixes for minimal DFAs is NL-complete.

Proof. It remains to prove NL-hardness for DFAs. Consider the construction from the proof of Theorem[17] and add a new accepting state, \( f \), and new transitions under fresh letters from \( q_0 \) to every node of \( G \), and from every node of \( G \) to \( f \). Transitions under these letters are undefined in \( B \). Then \( A \) and \( B \) are minimal DFAs and there exists an infinite tower of prefixes if and only if \( t \) is reachable from \( s \).

8. Conclusions

We have provided upper and lower bounds on the height of maximal finite towers between two regular languages represented by nondeterministic finite automata in the case there is no infinite tower. Both the upper and lower bound is exponential with respect to the size of the alphabet, which means that the algorithm of Section 4 needs to handle at least exponential number of regular languages. In addition, we have shown that the exponential lower bound can be obtained not only for NFAs, but also for DFAs. And the observation that the lower bounds for NFAs are formed by sequences of prefixes has motivated the investigation of towers of prefixes.

Finally, note that there is still a gap between the upper and lower bound on the height of towers. Moreover, the question how to efficiently compute a piecewise testable separator is open, including the question whether an efficient algorithm exists.

Acknowledgements

The authors would like to thank Stefan Borgwardt, Wojciech Czerwiński, Galina Jirásková, Markus Krötzsch, and Wim Martens for valuable discussions on various parts of the paper.

References

[1] Almeida, J., 1990. Implicit operations on finite J-trivial semigroups and a conjecture of I. Simon. Journal of Pure and Applied Algebra 69, 205–218.
[2] Almeida, J., 1995. Finite semigroups and universal algebra. Vol. 3 of Series in Algebra. World Scientific.
[3] Almeida, J., Costa, J. C., Zeitouni, M., 2008. Pointwise sets with respect to \( R \) and \( J \). Journal of Pure and Applied Algebra 212 (3), 486–499.
[4] Almeida, J., Zeitouni, M., 1997. The pseudovariety \( J \) is hyperdecomposable. RAIRO – Theoretical Informatics and Applications 31 (5), 457–482.
[5] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maier, E., Yergeau, F., November 2008. Extensible Markup Language XML 1.0 (fifth edition). Tech. rep., World Wide Web Consortium (W3C), W3C Recommendation, http://www.w3.org/TR/2008/REC-xml-20081126/.
[6] Cho, S., Huyng, D. T., 1991. Finite-automaton aperiodicity is PSPACE-complete. Theoretical Computer Science 88 (1), 99–116.
[7] Choffrut, C., D’Alessandro, F., Varricchio, S., 2007. On the separability of sparse context-free languages and of bounded rational relations. Theoretical Computer Science 381 (1-3), 274–279.
[8] Czerwiński, W., Martens, W., Masopust, T., 2013. Efficient separability of regular languages by subsequences and suffixes. In: Fomin, F. V., Freivalds, R., Kwiatkowska, M. Z., Peleg, D. (Eds.), Proc. of International Colloquium on Automata, Languages and Programming (ICALP). Vol. 7966 of Lecture Notes in Computer Science. Springer, pp. 150–161, full version available at http://arxiv.org/abs/1303.0966.
[9] Czerwiński, W., Martens, W., Masopust, T., 2013. Personal communication.
[10] Garey, M. R., Johnson, D. S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman.
[11] Greenlaw, R., Hoover, H. J., Ruzzo, W. L., 1995. Limits to Parallel Computation: P-Completeness Theory. Oxford University Press.
[12] Higman, G., 1952. Ordering by divisibility in abstract algebras. Proceedings of the London Mathematical Society s3-2 (1), 326–336.
[13] Holub, Š., Jirásková, G., Masopust, T., 2014. On upper and lower bounds on the length of alternating towers. In: Czumaj-Varju, E., Dietzfelbinger, M., Ńisk, Z. (Eds.), Proc. of Mathematical Foundations of Computer Science (MFCS). Vol. 8634 of Lecture Notes in Computer Science. Springer, pp. 315–326.
[14] Hunt III, B. H., 1982. On the decidability of grammar problems. Journal of the ACM 29 (2), 429–447.
[15] Karandikar, P., Niewerth, M., Schnoebelen, P., 2015. On the state complexity of closures and interiors of regular languages with subwords and superwords. Theoretical Computer Science. http://dx.doi.org/10.1016/j.tcs.2015.09.028, to appear.
[16] Kontorovich, L., Cortes, C., Mohri, M., 2006. Learning linearly separable languages. In: Balcázar, J. L., Long, P. M., Stephan, F. (Eds.), Proc. of Algorithmic Learning Theory (ALT). Vol. 4264 of Lecture Notes in Computer Science. Springer, pp. 288–303.
[17] Lawson, M., 2003. Finite Automata. Chapman and Hall/CRC.
[18] Losemann, K., Martens, W., 2012. The complexity of evaluating path expressions in SPARQL. In: Benedikt, M., Krötzsch, M., Lenzneri, M. (Eds.), Proc. of Principles of database systems (PODS). ACM, pp. 101–112.
[19] Martens, W., Neven, F., Niewerth, M., Schwentick, T., 2012. Developing and analyzing XSDs through BonXai. PVLDB 5 (12), 1994–1997.
[20] Papadimitriou, C. H., 1994. Computational Complexity. Addison-Wesley.
[21] Pérez, J., Arenas, M., Gutiérrez, C., 2010. nSPARQL: A navigational language for RDF. Journal of Web Semantics 8 (4), 255–270.
[22] Perrin, D., Pin, J.-E., 2003. Finite Automata. Chapman and Hall.
[23] Place, T., van Rooijen, L., Zeitoun, M., 2013. Separating regular languages by piecewise testable and unambiguous languages. In: Chatterjee, K., Sgall, J. (Eds.), Proc. of Mathematical Foundations of Computer Science (MFCS). Vol. 8087 of Lecture Notes in Computer Science. Springer, pp. 729–740.

[24] Place, T., Zeitoun, M., 2014. Going higher in the first-order quantifier alternation hierarchy on words. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (Eds.), Proc. of International Colloquium on Automata, Languages and Programming (ICALP). Vol. 8573 of Lecture Notes in Computer Science. Springer, pp. 342–353.

[25] Place, T., Zeitoun, M., 2014. Separating regular languages with first-order logic. In: Proc. of the joint meeting of the Conference on Computer Science Logic (CSL) and the Symposium on Logic in Computer Science (LICS). ACM, pp. 75:1–75:10.

[26] Rozenberg, G., Salomaa, A. (Eds.), 1997. Handbook of Formal Languages. Vol. 1–3. Springer, Berlin.

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

[28] Simon, I., 1972. Hierarchies of events with dot-depth one. Ph.D. thesis, Department of Applied Analysis and Computer Science, University of Waterloo, Canada.

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

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

[31] Stern, J., 1985. Characterizations of some classes of regular events. Theoretical Computer Science 35, 17–42.

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

[33] Trahtman, A. N., 2001. Piecewise and local threshold testability of DFA. In: Freivalds, R. (Ed.), Proc. of International Symposium on Fundamentals of Computation Theory (FCT). Vol. 2138 of Lecture Notes in Computer Science. Springer, pp. 347–358.