On Pumping Problems for Unary Regular Languages

Hermann Gruber[1], Markus Holzer[2], and Christian Rauch[2]

[1] Planerio GmbH, Theresienhöhe 11A, 80339 München, Germany
h.gruber@planerio.de
[2] Institut für Informatik, Universität Giessen
Arndtstr. 2, 35392 Giessen, Germany
{holzer,christian.rauch}@informatik.uni-giessen.de

Abstract. Recently, the descriptional and computational complexity of various pumping lemmata for general regular languages have been investigated in the literature. There it turned out that in almost all cases tight bounds on the operational complexity of minimal pumping constants for regular languages have been obtained. From the computational perspective it was shown that in most cases the question whether a certain value can serve as a pumping constant w.r.t. a fixed pumping lemma is computationally intractable. Whether similar results can be obtained for restricted regular languages, such as unary regular languages, was left open—a language is unary if the underlying alphabet is a singleton set. Here we fill this gap by considering in detail questions on various pumping lemmata for unary regular languages. While some of the results obtained are similar to those in the general case, we also find significant differences. The results presented here fit well with the previous results and give a mostly complete picture of the problems in question.

1 Introduction

To pump, or not to pump, that is the question, if one tries to prove that a specific language is *not* regular. Pumping lemmata are a main tool in any formal language and automata course for this task—see, [?] for a comprehensive survey of different variants of pumping lemmata for regular, context-free languages and beyond. One of the most common pumping lemmata, yet not the first one [?] of its kind, can be found in [?, page 70, Theorem 11.1].

Lemma 1. *Let L be a regular language over Σ. Then, there is a constant p (depending on L) such that the following holds: If $w \in L$ and $|w| \geq p$, then there are words $x \in \Sigma^*$, $y \in \Sigma^+$, and $z \in \Sigma^*$ such that $w = xyz$ and $xy^tz \in L$ for all $t \geq 0$—it is then said that y can be* pumped *in w.*

Most pumping lemmata describe a necessary condition for a language to be regular, as the above one, but there are also lemmata that characterize the regular languages, by describing a necessary and sufficient condition for languages to be regular. One of the first ones of this kind is attributed to Jaffe [?] and reads as follows:

Lemma 2. *A language L is regular if and only if there is a constant p (depending on L) such that the following holds: If $w \in \Sigma^*$ and $|w| = p$, then there are words $x \in \Sigma^*$, $y \in \Sigma^+$, and $z \in \Sigma^*$ such that $w = xyz$ and*[3]

$$wv = xyzv \in L \iff xy^tzv \in L$$

for all $t \geq 0$ and each $v \in \Sigma^*$.

Recently, the descriptional and computational complexity of problems related to some specific pumping lemmata were considered in a series of papers [**?**,**?**,**?**,**?**,**?**,**?**]. Roughly speaking this is the study of the minimal pumping constants that satisfy a particular pumping lemma, such as one of the above stated ones, from the aforementioned complexity perspectives. Let $\mathtt{mpc}(L)$ ($\mathtt{mpe}(L)$, respectively) denote the smallest number p that satisfies the condition of Lemma **??** (Lemma **??**, respectively) when considering the regular language L. We state one example result for a descriptional complexity problem from [**?**] and another one for a computational problem [**?**]: first consider the complementation operation w.r.t. minimal pumping constants. Starting with a language L over the alphabet Σ satisfying $\mathtt{mpc}(L) = n$, one ends up with

$$\mathtt{mpc}(\Sigma^* \setminus L) \in \begin{cases} \{1\}, & \text{if } n = 0, \\ \mathbb{N}_0 \setminus \{1\}, & \text{if } n = 1, \\ \mathbb{N}, & \text{otherwise.} \end{cases}$$

For $\mathtt{mpe}$ the situation is easier, since $\mathtt{mpe}(L) = \mathtt{mpe}(\Sigma^* \setminus L)$. Second, asking the question whether for a language given by a finite automaton and a particular value of p the properties of a pumping lemma is satisfied results in the PUMPING-PROBLEM. For DFAs this problem is coNP-complete regardless whether Lemma **??** or Lemma **??** is considered. For NFAs the situation changes to coNP-hardness and containment in the second level of the polynomial hierarchy for Lemma **??**, while becoming PSPACE-complete for Lemma **??**. All these results were proven for languages over an alphabet of at least two letters. Thus, the question arises, which results still hold true for unary regular languages? We partially solve this question in this paper in the affirmative. Observe, that the pumping condition from Lemma **??** simplifies to

> "If $w \in L$ and $|w| \geq p$, then there are words $x \in \Sigma^*$ and $y \in \Sigma^+$ such that $w = xy$ and $xy^t \in L$ for $t \geq 0$,"

for unary regular languages, since concatenation is commutative for unary languages. A similar simplification applies to Lemma **??** when considering unary regular languages. Thus, in both cases the decomposition of the words becomes easier, which may affect some of the known pumping problem results for languages over alphabets that are at least binary—see the summary of results shown in the Tables **??** and **??** on page **??** and **??**, respectively. Due to space constraints most proofs are omitted.

[3] Observe that the words $w = xyz$ and xy^tz, for all $t \geq 0$, belong to the same Myhill-Nerode equivalence class of the language L. Thus, one can say that the pumping of the word y in w *respects equivalence classes*.

2 Preliminaries

We assume the reader to be familiar with the basics in computational complexity theory [?]. In particular we recall the inclusion chain: $\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}$. Here P (NP, respectively) denotes the class of problems solvable by deterministic (nondeterministic, respectively) Turing machines in polytime, and PSPACE refers to the class of languages accepted by deterministic or nondeterministic Turing machines in polynomial space [?]. As usual, the prefix co refers to the complement class. For instance, coNP is the class of problems that are complements of NP problems. Moreover, recall the complexity class Π_2^{P} from the polynomial hierarchy, which can be described by polynomial time bounded oracle Turing machines. Here $\Pi_2^P = \mathsf{coNP}^{\mathsf{NP}}$, where $\mathsf{coNP}^{\mathsf{A}}$ is the set of decision problems solvable in polynomial time by a universal Turing machine with an oracle for some language in the class A. The class NP^{A} is defined analogously on a universal Turing machine. Moreover, let $\Theta_2^{\mathsf{P}} = \mathsf{P}_{||}^{\mathsf{NP}}$, which is the set of all decision problems that can be solved by a deterministic Turing machine in polynomial time with access to an NP oracle such that a list of all queries is formed before any of them is made (non-adaptive queries). The inclusion chain $\mathsf{coNP} \subseteq \Theta_2^{\mathsf{P}} \subseteq \Pi_2^{\mathsf{P}}$ is known [?]. Completeness and hardness are always meant with respect to deterministic many-one logspace reducibilities ($\leq_m^{\log}$) unless stated otherwise.

Next we fix some definitions on finite automata—cf. [?]. A *nondeterministic finite automaton* (NFA) is a quintuple $A = (Q, \Sigma, \cdot, q_0, F)$, where Q is the finite set of *states*, Σ is the finite set of *input symbols*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states*, and the *transition function* $\cdot$ maps $Q \times \Sigma$ to 2^Q. Here 2^Q refers to the powerset of Q. The *language accepted* by the NFA A is defined as $L(A) = \{\, w \in \Sigma^* \mid (q_0 \cdot w) \cap F \neq \emptyset \,\}$, where the transition function is recursively extended to a mapping $Q \times \Sigma^* \to 2^Q$ in the usual way. An NFA A is said to be *partial deterministic* if $|q \cdot a| \leq 1$ and *deterministic* (DFA) if $|q \cdot a| = 1$ for all $q \in Q$ and $a \in \Sigma$. In these cases we simply write $q \cdot a = p$ instead of $q \cdot a = \{p\}$. Note that every partial DFA can be made complete by introducing a non-accepting sink state that collects all non-specified transitions. For a DFA, obviously every letter $a \in \Sigma$ induces a mapping from the state set Q to Q by $q \mapsto q \cdot a$, for every $q \in Q$. Finally, a finite automaton is *unary* if the input alphabet Σ is a singleton set, that is, $\Sigma = \{a\}$, for some input symbol a.

The *deterministic state complexity of a finite automaton* A with state set Q is referred to as $\mathtt{sc}(A) := |Q|$ and the *deterministic state complexity of a regular language* L is defined as

$$\mathtt{sc}(L) = \min\{\, \mathtt{sc}(A) \mid A \text{ is a DFA accepting } L \text{, i.e., } L = L(A) \,\}.$$

A similar definition applies for the *nondeterministic state complexity of a regular language* by changing DFA to NFA in the definition, which we refer to as $\mathtt{nsc}(L)$. It is well known that

$$\mathtt{nsc}(L) \leq \mathtt{sc}(L) \leq 2^{\mathtt{nsc}(L)},$$

for every regular language L.

A finite automaton is *minimal* if its number of states is minimal with respect to the accepted language. It is well known that each minimal DFA is isomorphic to the DFA induced by the Myhill-Nerode equivalence relation. The *Myhill-Nerode* equivalence relation $\sim_L$ for a language $L \subseteq \Sigma^*$ is defined as follows: for $u, v \in \Sigma^*$ let $u \sim_L v$ if and only if $uw \in L \iff vw \in L$, for all $w \in \Sigma^*$. The equivalence class of u is referred to as $[u]_L$ or simply $[u]$ if the language is clear from the context and it is the set of all words that are equivalent to u w.r.t. the relation $\sim_L$, i.e., $[u]_L = \{\, v \mid u \sim_L v \,\}$. Therefore we refer to the automaton induced by the Myhill-Nerode equivalence relation $\sim_L$ as the minimal DFA for the language L. On the other hand there may be minimal non-isomorphic NFAs for L.

3 Results on Pumping for Unary Finite Automata

At first we summarize some basic properties on the minimal pumping constants w.r.t. Lemmata **??** and **??** for arbitrary regular languages, not necessarily unary languages. The following relations

$$\texttt{mpc}(L) \leq \texttt{mpe}(L) \leq \texttt{sc}(L) \quad \text{and} \quad \texttt{mpc}(L) \leq \texttt{nsc}(L)$$

for every regular language L are known from [?,?,?]; interestingly the two measures mpe and nsc are incomparable [?]. Further simple facts for a regular language $L \subseteq \Sigma^*$ are the following:

1. $\texttt{mpc}(L) = 0$ if and only if $L = \emptyset$,
2. for every nonempty finite language L we have

$$\texttt{mpc}(L) = 1 + \max\{\, |w| \mid w \in L \,\} \leq \texttt{mpe}(L) \leq 2 + \max\{\, |w| \mid w \in L \,\},$$

3. $\texttt{mpc}(L) = 1$ implies that the empty word λ is in L, and
4. $\texttt{mpe}(L) = 1$ if and only if $L = \emptyset$ or $L = \Sigma^*$.

By definition of the minimal pumping constant w.r.t. Lemma **??** we know that for every regular language L there is a nonempty word $w \in L$ with $|w| = \texttt{mpc}(L) - 1$. A similar result is *not* valid for the minimal pumping constant w.r.t. Lemma **??** since one cannot force $w \in L$ in this case. The following observation is immediate by Jaffe's proof, cf. [?], and was mentioned first in [?]—in contrast, there is *no* obvious relation between mpc and the longest path in the finite state device.

Lemma 3. *Let A be a DFA and $L := L(A)$. Then $\texttt{mpe}(L) \leq \ell_A + 1$, where ℓ_A is the length, i.e., number of transitions, of the longest path of the automaton A. If L is a unary language, then $\texttt{mpe}(L) = \texttt{sc}(L)$.*

Thus, the descriptional complexity measures mpe and sc are equivalent for unary finite languages.

3.1 Descriptional Complexity of the Operation Problem

This section is devoted to the descriptional complexity of the operation problem w.r.t. minimal pumping constants for the pumping lemmata introduced here. In general, the operation problem for a regularity preserving n-ary function $\circ$ on languages and a descriptional complexity measure K is given by $g_\circ^K(k_1, k_2, \ldots, k_n)$ as the set of all numbers k such that there are regular languages $L_1, L_2, \ldots, L_n$ with descriptional complexity $K(L_i) = k_i$, for $1 \leq i \leq n$, and $K(\circ(L_1, L_2, \ldots, L_n)) = k$. If only unary regular languages are taken into account, then we write $g_\circ^{K,u}(k_1, k_2, \ldots, k_n)$ for the operational complexity of the $\circ$-operation. Here we assume that $K \in \{\texttt{mpc}, \texttt{mpe}\}$. Results on the descriptional complexity of the measures `mpc`, `mpe`, and others can be found in [?,?], but only the operation problem for `mpc` and some other measures, except for `mpe`, was investigated in more detail in [?]—the operation problem is completely untouched for `mpe` yet. The behaviour of the `mpc` measure differs with respect to finiteness/infinity of ranges depending on the considered operation. Our findings are summarized in Table **??**, where the set of all natural numbers not including zero is denoted by $\mathbb{N}$; if zero is included, then we write $\mathbb{N}_0$ instead. Moreover, for $n \geq 2$, let $\mathbb{N}_{\geq n}$ refer to the set $\{n, n+1, \ldots\}$. The gray shaded entries in the table are new results.

	Minimal pumping constant		
	`mpc`		`mpe`
Operation	unary	general	unary
Reversal	$\{n\}$	$\{n\}$	$\{n\}$
Prefix	$\{0\}$, if $n = 0$, $\{1, n\}$, otherwise.	$\{0\}$, if $n = 0$, $\{1, 2, \ldots, n\}$, otherwise.	$\{0\}$, if $n = 0$, $\{1, n\}$, otherwise.
Suffix	$\{0\}$, if $n = 0$, $\{1, n\}$, otherwise.	$\{0\}$, if $n = 0$, $\{1, 2, \ldots, n\}$, otherwise.	$\{0\}$, if $n = 0$, $\{1, n\}$, otherwise.
Complement	$\{1\}$, if $n = 0$, $\mathbb{N}_0 \setminus \{1\}$, if $n = 1$, $\mathbb{N} \setminus \{n\}$, otherwise.	$\{1\}$, if $n = 0$, $\mathbb{N}_0 \setminus \{1\}$, if $n = 1$, $\mathbb{N}$, otherwise.	$\{n\}$
Intersection	$\{0\}$, if $m = 0$ or $n = 0$, $\{1\}$, if $m = n = 1$, $\mathbb{N}_{\geq n}$, if $m = n \geq 2$, $\mathbb{N}_0$, otherwise.	$\{0\}$, if $m = 0$ or $n = 0$, $\mathbb{N}_0 \setminus \{2\}$, if $m = n = 1$, $\mathbb{N}_0$, otherwise.	$[1, (\lfloor n/2 \rfloor - 1) \cdot \lfloor n/4 \rfloor - 1] \subseteq \cdot$ $\cdot \cap [n^2 - n + 2, n^2] = \emptyset$
Union	$\{0\}$, if $m = 0$ or $n = 0$, $\{1\}$, if $m = n = 1$, $\mathbb{N}_{\geq n}$, if $m = n \geq 2$, $\mathbb{N}_0$, otherwise.	$\max\{m, n\}$, if $m = 0$ or $n = 0$, $\{1, 2, \ldots, \max\{m, n\}\}$, otherwise.	$[1, (\lfloor n/2 \rfloor - 1) \cdot \lfloor n/4 \rfloor - 1] \subseteq \cdot$ $\cdot \cap [n^2 - n + 2, n^2] = \emptyset$

Table 1. Descriptional complexity of the operation problem for the measures `mpc` and `mpe`. The `mpe`-results for intersection and union are only valid for $m = n$. An upper bound is proved in the paper for general m and n for both cases. Gray shaded entries indicate new results.

Before we start our investigation of the operation problem in detail, we state the following auxiliary lemma, which turns out to be quite useful in the forthcoming arguments, since it is a lower bound argument on the minimal pumping constant w.r.t. Lemma **??**. The statement is a reformulation of the fact that a word cannot be pumped. Thus, we omit the straightforward proof.

Lemma 4. *Let L be a unary regular language over $\Sigma = \{a\}$. Then $\mathtt{mpc}(L) > \ell$ if there is a word $w = a^{\ell}$ in L such that*

1. *there is* no *word $a^k \in L$, for $0 \leq k < \ell$, or*
2. *the property $a^k(a^{\ell-k})^* \not\subseteq L$ holds, for every word $a^k \in L$ with $k < \ell$.*

Let us start with the reversal operation. As usual the reversal of a word w is denoted by w^R and the reversal of the language L by L^R. For unary languages L we have $L = L^R$ which implies the following theorem—compare with the general result for $\mathtt{mpc}$ listed in Table **??**.

Theorem 5. $g_R^{K,u}(n) = \{n\}$*, for $K \in \{\mathtt{mpc}, \mathtt{mpe}\}$.*

The next two operations we consider are the closure under prefix and suffix, denoted by Pref(.) and Suff(.), respectively. The descriptional complexity of both measures behaves in the same way for these operations.

Theorem 6. *For $K \in \{\mathtt{mpc}, \mathtt{mpe}\}$ we have*

$$g_{\mathrm{Pref}}^{K,u}(n) = g_{\mathrm{Suff}}^{K,u}(n) = \begin{cases} \{0\}, & \textit{if } n = 0, \\ \{1, n\}, & \textit{otherwise.} \end{cases}$$

We continue our investigation with the complement operation. Let Σ be an alphabet and L be a language over Σ, then we refer to the complement of L as $C_\Sigma(L) = \Sigma^* \setminus L$. Here we find the following situation for unary languages—see Table **??**. The results for the two measures are entirely different. In the next and the forthcoming proofs we use the abbreviation $L^{\leq n}$ ($L^{\geq n}$, respectively), to refer to the set $\bigcup_{i=0}^{n} L^i$ ($\bigcup_{i=n}^{\infty} L^i$, respectively), for any language L.

Theorem 7. *It holds*

$$g_{C_\Sigma}^{\mathtt{mpc},u}(n) = \begin{cases} \{1\}, & \textit{if } n = 0, \\ \mathbb{N}_0 \setminus \{1\}, & \textit{if } n = 1, \\ \mathbb{N} \setminus \{n\}, & \textit{otherwise.} \end{cases} \quad \textit{and} \quad g_{C_\Sigma}^{\mathtt{mpe},u}(n) = \{n\}.$$

Proof. We consider first the measure $\mathtt{mpc}$. Like in [?], the high-level strategy is, for given integers n and k, either to find a witness language $L = L_{n,k}$ such that $\mathtt{mpc}(L) = n$ and $\mathtt{mpc}(C_\Sigma(L)) = k$, or to prove that no such witness language exists.

We distinguish the cases $n = 0$, $n = 2 = k + 1$, $n = k \geq 1$, and $n \geq 3$ with $n > k$—all other cases can be derived by using $C_\Sigma(C_\Sigma(L)) = L$:

1. For $n = 0$ we know that $L = \emptyset$ which implies that $C_\Sigma(L) = \Sigma^*$ for all alphabets Σ^* which implies $\mathtt{mpc}(C_\Sigma(L)) = 1$.
2. In the case $n = 2 = k + 1$ we set $L = a^+$ which fulfills $\mathtt{mpc}(L) = 2$ since the word a is the only non-pumpable word in L. On the other hand $C_\Sigma(L) = \{\lambda\}$ is a finite language and therefore satisfies $\mathtt{mpc}(C_\Sigma(L)) = 1$.

3. For $n = k \geq 1$ assume that $\texttt{mpc}(L_{n,k}) = \texttt{mpc}(C_\Sigma(L_{n,k})) = k = n$. We observe that due to the Lemma **??** there must be a word $w \in L_{n,k}$ such that $|w| = \texttt{mpc}(L) - 1 = n - 1$ which in turn implies that $w = a^{n-1}$. Analogously we obtain that $w = a^{n-1} \in C_\Sigma(L_{n,k})$ and therefore $w = a^{n-1} \in L_{n,k} \cap C_\Sigma(L_{n,k}) = \emptyset$ which is a contradiction.
4. If $n \geq 3$ and $n > k$ we set
$$L_{n,k} = \{a\}^{\leq n-1} \setminus (\{a^{k-1}\}) \quad \text{and thus} \quad C_\Sigma(L_{n,k}) = \{a^{k-1}\} \cup \{a\}^{\geq n}.$$
Since $L_{n,k}$ is a finite language we have $\texttt{mpc}(L_{n,k}) = n$. On the other hand we observe that each word in $C_\Sigma(L_{n,k})$ is pumpable except the word a^{k-1} which is the shortest word in $C_\Sigma(L_{n,k})$. Therefore we obtain $\texttt{mpc}(C_\Sigma(L_{n,k})) = k$.

Finally, the statement for $\texttt{mpe}$ follows directly from Lemma **??**. □

Now we come to the intersection of two unary languages.

Theorem 8. *We have*
$$g_\cap^{\texttt{mpc},u}(m,n) = \begin{cases} \{0\}, & \textit{if } m = 0 \textit{ or } n = 0, \\ \{1\}, & \textit{if } m = n = 1, \\ \mathbb{N}_{\geq n}, & \textit{if } m = n \geq 2, \\ \mathbb{N}_0, & \textit{otherwise.} \end{cases}$$

For the minimal pumping constants w.r.t. Lemma **??** the situation concerning the intersection operation is much more involved. First recall that for languages L defined over arbitrarily large alphabets we have $\texttt{mpe}(L) \leq \texttt{sc}(L)$ and in particular $\texttt{mpe}(L) = \texttt{sc}(L)$ for unary languages L. Thus, the results on unary languages presented below can be rewritten using $\texttt{sc}$ instead of $\texttt{mpe}$ and they remain still valid.

The cross-product construction can be used to determine the automaton accepting the intersection of two regular languages given by automata. Thus, $g_\cap^{\texttt{mpe},u}(m,n) \subseteq [1, mn]$ and by a result from [**?**, Lemma 1] we also have the inclusion relation $[1,m] \subseteq g_\cap^{\texttt{mpe},u}(m,n)$, if $1 \leq m \leq n$. Before we can make it more explicit which values in the interval $[1, mn]$ can be reached, we need some notation. For two sets S_1 and S_2 of numbers from $\mathbb{N}_0$, let
$$S_1 \oplus S_2 := \{\, x + y \mid x \in S_1 \text{ and } y \in S_2 \,\}$$
denote their Minkowski addition. For $m, n \geq 2$ and $k \geq 1$ we define
$$M_{m,n} = \{\, t_1 t_2 \mid m \bmod t_1 = n \bmod t_2 = 0 \,\}, \quad \text{and} \quad M_{1,k} = M_{k,1} = \{1, k\}.$$
Now we are ready for the next theorem, which is a consequence of a result from [**?**].

Theorem 9. *We have*
$$g_\cap^{\texttt{mpe},u}(m,n) \subseteq \bigcup_{\ell=1}^{m} \bigcup_{k=1}^{n} (M_{k,\ell} \oplus [0, \max\{n-k, m-\ell\}]).$$

The theorem above gives us a rather implicit description of the values that can be possibly obtained by the intersection of two unary regular languages. In order to get a better impression of this result we list the complements of the upper bound sets described in Theorem **??** for small values of m and n in Table **??**—the complementation is done w.r.t. to the sets $[1, mn]$. For instance, for $m = 6$ and

	n					
m	2	3	4	5	6	7
2	$\emptyset$					
3	$\emptyset$	$\{8\}$				
4	$\emptyset$	$\{11\}$	$\{11, 14, 15\}$			
5	$\emptyset$	$\{14\}$	$\{18, 19\}$	$\{18, 19, 22, 23, 24\}$		
6	$\emptyset$	$\{17\}$	$\{22, 23\}$	$\{23, 27, 28, 29\}$	$\{23, 27, 28, 29, 32, 33, 34, 35\}$	
7	$\emptyset$	$\{20\}$	$\{23, 26, 27\}$	$\{32, 33, 34\}$	$\{33, 34, 38, 39, 40, 41\}$	$\{33, 34, 38, 39, 40, 41, 44, 45, 46, 47, 48\}$

Table 2. The sets $[1, mn] \setminus \bigcup_{\ell=1}^{m} \bigcup_{k=1}^{n} (M_{k,\ell} \oplus [0, \max\{n-k, m-\ell\}])$ for small m and n.

$n = 5$ we find that the values in the set $\{23, 27, 28, 29\}$ cannot be achieved as $\mathtt{mpe}$-values by the intersection operation on unary regular languages with $\mathtt{mpe}$-complexity m and n, respectively. A closer look reveals that only numbers from the upper range close to the rightmost border of the interval $[1, mn]$ are listed. Since the analysis of the function $g_{\cap}^{\mathtt{mpe},u}$ in general is quite involved, we continue our investigation with a focus on the case $m = n$.

Consequently, we are interested in the function $g_{\cap}^{\mathtt{mpe},u}(n, n) \subseteq [1, n^2]$. First, we show that the upper range $[n^2 - n + 2, n^2]$ cannot be reached by $g_{\cap}^{\mathtt{mpe},u}(n, n)$, for large enough n. From Theorem **??** we can derive that the upper range of possible numbers is not attainable—the stated result is also a direct implication of [**?**, Theorem 4].

Lemma 10. *For $n \geq 2$ we have $g_{\cap}^{\mathtt{mpe},u}(n, n) \cap [n^2 - n + 2, n^2] = \emptyset$.*

Proof. Observe that if L and L' are cyclic, i.e., they are accepted by a DFA which consists of one cycle with n states and an empty tail, then $L \cap L'$ is accepted by a DFA consisting of one cycle with n states and an empty tail, too. One easily observes that if either L or L' is not cyclic, then within the cross-product construction we obtain an automaton accepting $L \cap L'$ with less than n^2 states. Hence, the number n^2 cannot be in $g_{\cap}^{\mathtt{mpe},u}(n, n)$. On the other hand we have that the greatest number in

$$\bigcup_{\ell=1}^{n} \bigcup_{k=1}^{n} (M_{k,\ell} \oplus [0, \max\{n-k, n-\ell\}]) \setminus \{n^2\}$$

is $n \cdot (n-1) + 1 = n^2 - n + 1$ which proves the stated claim with Theorem **??**. $\square$

Regarding the lower end of the interval, the next theorem improves the aforementioned result $[1,n] \subseteq g_{\cap}^{\mathtt{mpe},u}(n,n)$ from [?, Lemma 1] to a much larger range:

Theorem 11. *For $n \geq 2$ we have $[1, (\lfloor n/2 \rfloor - 1) \cdot \lfloor n/4 \rfloor - 1] \subseteq g_{\cap}^{\mathtt{mpe},u}(n,n)$.*

With the aid of Theorems **??** and **??** we can tell for each number whether it can be reached or not *via* intersection, except for the numbers in the range

$$[(\lfloor n/2 \rfloor - 1) \cdot \lfloor n/4 \rfloor, n^2 - n + 1].$$

By computing the numbers $g_{\cap}^{\mathtt{mpe},u}(n,n)$, using exhaustive computer search, up to the value $n = 10$, we found that numbers which are not covered by the above theorems satisfy a specific side condition—we may require that the factors in the products in $M_{k,\ell}$ are coprime. Therefore we conjecture the following.

Conjecture 1 *We have $g_{\cap}^{\mathtt{mpe},u}(n,n) = \bigcup_{\ell=1}^{n} \bigcup_{k=1}^{n} (\widehat{M_{k,\ell}} \oplus [0, \max\{n-k, n-\ell\}])$, where*

$$\widehat{M_{k,\ell}} = \begin{cases} \{1, \max\{k,\ell\}\}, & \text{if } k = 1 \text{ or } \ell = 1, \\ \{\, t_1 t_2 \mid k \bmod t_1 = \ell \bmod t_2 = 0, \gcd(t_1,t_2) = 1 \,\}, & \text{otherwise.} \end{cases}$$

Aside from this, for the union operation on regular languages, we can apply De Morgan's law to derive the following statement from Theorems **??** and **??**.

Theorem 12. *For $n \geq 2$ we have*

$$[1, (\lfloor n/2 \rfloor - 1) \cdot \lfloor n/4 \rfloor - 1] \subseteq g_{\cup}^{\mathtt{mpe},u}(n,n)$$

and

$$g_{\cup}^{\mathtt{mpe},u}(n,n) \cap [n^2 - n + 2, n^2] = \emptyset.$$

3.2 Computational Complexity of the Pumping-Problem

We will consider the following decision problem [?] related to the pumping lemmata stated in the introduction:

Language-Pumping-Problem or, for short, Pumping-Problem:
Input: a finite automaton A and a natural number p, i.e., an encoding $\langle A, 1^p \rangle$.
Output: Yes, if and only if the statement from Lemma **??** holds for the language $L(A)$ w.r.t. the value p.

For DFAs and NFAs the following results are known: already for DFAs this problem is intractable, namely coNP-complete [?], regardless whether we check for Kozen's [?] or Jaffe's [?] pumping property. This is quite remarkable, since it is a rare example of a computationally intractable property of a given *deterministic* finite automaton. The latter pumping property turned out to be more complex for NFAs, namely PSPACE-complete, while the former was shown to be coNP-hard and contained in Π_2^{P}, the second co-level of the polynomial hierarchy, for nondeterministic finite state devices [?]. Moreover, for NFAs, and even for DFAs, inapproximability results were recently shown for both pumping properties assuming the Exponential Time Hypothesis (ETH) [?,?]. In all cases the involved finite automata were at least binary. Thus, the question arises, whether similar results on the Pumping-Problem also hold for unary finite state devices. We answer this question in the affirmative. Our findings are summarized in Table **??**.

	Pumping-Problem w.r.t. ...	
Automaton	Lemma **??**	Lemma **??**
unary DFA	L-complete	L-complete
DFA	coNP-complete	coNP-complete
unary NFA	coNP-hard in Π_2^{P}	coNP-hard in Θ_2^{P}
NFA	coNP-hard in Π_2^{P}	PSPACE-compl.

Table 3. Complexity of the Pumping-Problem for variants of finite state devices. Gray shaded entries indicate new results.

We first prove an auxiliary result on the relation of minimal pumping constants and the universality of unary languages:

Lemma 13. *Let $L \subseteq \{a\}^*$ be a unary regular language. Then the following statements hold:*

1. *If the empty word λ and the word a are in L, then $\mathtt{mpc}(L) = 1$ iff $L = a^*$.*
2. *If λ is in L, then $\mathtt{mpe}(L) = 1$ if and only if $L = a^*$.*

Let us first focus on the Pumping-Problem for unary DFAs. The following lemma is quite useful for the study of this problem. Recall that the condition $xy^tz \in L$ simplifies to $xy^t \in L$ since concatenation is commutative for unary languages.

Lemma 14. *Let $A = (Q, \{a\}, \cdot, q_0, F)$ be a unary DFA and $w = xyz$ a word in $L(A)$ with $x \in a^*$ and $y \in a^+$. Then $xy^* \subseteq L(A)$ if and only if every word $x, xy, xy^2, \ldots, xy^\ell$ belongs to $L(A)$, for $\ell = |Q|$.*

Now we are ready for the complexity of the Pumping-Problem for unary DFAs w.r.t. Lemma **??**.

Theorem 15. *Given a unary DFA A and a natural number p, it is* L*-complete w.r.t. weak reductions to decide whether for the language $L(A)$ the statement from Lemma* **??** *holds for the value p.*

We turn our attention to the Pumping-Problem for unary DFAs w.r.t. Lemma ??. Here we observe that the problem remains L-complete. However, we need an auxiliary result checking for the Myhill-Nerode classes that appear during pumping—compare with a similar result for arbitrary DFAs and NFAs recently shown in [?].

Lemma 16. *Given a unary DFA $A = (Q, \{a\}, \cdot, q_0, F)$ and two unary words x and y, deciding whether every word in xy^* describes the same equivalence class w.r.t. the Myhill-Nerode relation $\sim_{L(A)}$, is L-complete. If the automaton A is a unary NFA, the problem becomes coNP-complete.*

The following theorem gives the answer to the computational complexity of the Pumping-Problem for unary deterministic finite automata.

Theorem 17. *Given a unary DFA A and a natural number p, it is L-complete to decide whether for the language $L(A)$ the statement from Lemma ?? holds for the value p.*

Next we study the Pumping-Problem for unary NFAs. The non-universality problem for unary NFAs automata was shown to be NP-complete [?]. The classic reduction is from 3SAT, and makes use of the Chinese Remainder Theorem. The construction can be adapted to show our next inapproximability result under the assumption of the so-called *Exponential Time Hypothesis (ETH) [?, ?]:* there is no algorithm that solves 3-SAT in time $O^*(2^{o(n+m)})$, where n and m are the number of variables and clauses, respectively.

Theorem 18. *Let A be a unary NFA with s states. Then it is impossible to approximate the pumping constant with respect to Lemma ?? within a factor of $o(\sqrt[4]{s \log s})$ if the running time is in $2^{o\left(\sqrt[4]{\frac{s}{(\log s)^3}}\right)}$, assuming the Exponential Time Hypothesis. For the pumping constant with respect to Lemma ??, the inapproximability factor is even $2^{o\left(\sqrt[4]{s(\log s)^3}\right)}$ for the same running time, again assuming the Exponential Time Hypothesis.*

We note that a more efficient subexponential-time reduction for the NFA universality problem is given in [?], which might yield an improved lower bound on inapproximability. Nevertheless, from the previous proof we can deduce an coNP-lower bound for the Pumping-Problem for unary NFAs, regardless whether we consider Lemma ?? or ??. For the upper bound we explore an auxiliary statement that was shown in [?] and reads as follows:

Lemma 19. *Given an NFA $A = (Q, \Sigma, \cdot, q_0, F)$ and a word w over Σ, the language inclusion problem for w^* in $L(A)$ is coNP-complete.*

Now we are ready for our last theorem:

Theorem 20. *Given a unary NFA A and a natural number p, it is coNP-hard and it can be decided in Π_2^{P} whether for the language $L(A)$ the statement from Lemma ?? holds for the value p. In case Lemma ?? has to be fulfilled, the problem can be decided in Θ_2^{P}.*

Proof. Due to space constraints, we only give proofs for the upper bounds. When considering Lemma **??**, the upper bound is seen as follows—and literally is taken from [**?**]: Let $\langle A, p\rangle$ be an input instance of the problem in question, where Q is the state set of A. We construct a coNP Turing machine M with access to a coNP oracle: first the device M deterministically verifies whether $p \geq |Q|$, and if so, it halts and accepts. Otherwise the computation universally guesses ($\forall$-states) a word w with $p \leq |w| < |Q|$. On that particular branch M checks deterministically if w belongs to $L(A)$. If this is *not* the case the computation halts and accepts. Otherwise, M deterministically cycles through all valid decompositions $w = xyz$ with $|y| \geq 1$. Then it constructs a finite automaton B accepting the language quotient $(x^{-1} \cdot L(A)) \cdot z^{-1}$. Here, if A is deterministic, then so is B. Then M decides whether $y^* \subseteq L(B)$ with the help of the coNP oracle—compare Lemma **??**. If $y^* \subseteq L(B)$, then the cycling through the valid decompositions is stopped, and the device M halts and accepts. Notice that the latter is the case iff $xy^*z \subseteq L(A)$. Otherwise, i.e., if $y^* \not\subseteq L(B)$, the Turing machine M continues with the next decomposition in the enumeration cycle. Finally, if the cycle computation finishes, the Turing machine halts and rejects, because no valid decomposition of w was found that allows for pumping. In summary, the Turing machine operates universally, runs in polynomial time, and uses a coNP oracle. Thus, the containment in Π_2^{P} follows.

When considering Lemma **??**, we cannot apply this algorithm: it would not result in a polynomial time bound, since $|w| \leq 2^{|Q|}$ has to be satisfied. Thus, we proceed differently, and utilize the relation between the deterministic state complexity of a unary regular language and the minimal pumping constant w.r.t. Jaffe's pumping lemma as mentioned in Lemma **??**: if L is a unary regular language, then $\mathtt{mpe}(L) = \mathtt{sc}(L)$. Thus, we construct a P Turing machine M with non-adaptive access to an NP oracle that checks for inequivalence on unary finite state devices—recall that the value p from the input is encoded in unary: first M deterministically lists all words a^m that belong to $L(A)$ with $m \leq q$ by simulating the given NFA A with an incremental powerset construction. The words from this list will be used to determine the accepting states of the DFAs constructed next. Every unary DFA consists of a tail and a loop of states. The Turing machine then lists all unary DFAs by cycling through all combinations of tail and loop states such that the overall number of states does not exceed q. Here the above constructed list of words is used to assign the accepting states to these automata appropriately. By simple combinatorics one observes, that there exists only polynomially many unary DFAs that can be constructed in that way. Then the NP oracle is questioned on this list of unary DFAs. Since we are asking for inequivalence, the Turing machine halts and accepts if at least one query is answered negatively; otherwise the Turing machine halts and rejects. It is easy to see that M decides the PUMPING-PROBLEM w.r.t. Jaffe's pumping lemma. To summarize, the Turing machine operates deterministically, runs in polynomial time, and uses an NP oracle such that a list of all queries is formed before any of them is made. Thus, the containment within Θ_2^{P} follows. □

References

1. Cygan, M., Fomin, F., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms, chap. Lower Bounds Based on the Exponential-Time Hypothesis, pp. 467–521. Springer (2015). https://doi.org/10.1007/978-3-319-21275-3_14
2. Dassow, J., Jecker, I.: Operational complexity and pumping lemmas. Acta Inform. **59**, 337–355 (2022). https://doi.org/10.1007/s00236-022-00431-3
3. Fernau, H., Krebs, A.: Problems on finite automata and the exponential time hypothesis. Algorithms **10**(1), 24 (2017). https://doi.org/10.3390/a10010024
4. Gruber, H., Holzer, M., Rauch, C.: The pumping lemma for regular languages is hard. In: Nagy, B. (ed.) Proceedings of the 27th International Conference on Implementation and Application of Automata. pp. 128–140. No. 14151 in LNCS, Springer, Famagusta, Cyprus (2023). https://doi.org/10.1007/978-3-031-40247-0_9
5. Gruber, H., Holzer, M., Rauch, C.: On pumping preserving homomorphisms and the complexity of the pumping problem (extended abstract). In: Fazekas, S. (ed.) Proceedings of the 28th International Conference on I mplementation and Application of Automata. pp. 153–165. No. 15015 in LNCS, Springer, Akita, Japan (2024). https://doi.org/10.1007/978-3-031-71112-1_11
6. Gruber, H., Holzer, M., Rauch, C.: The pumping lemma for context-free languages is undecidable. In: Day, J.D., Manea, F. (eds.) Proceedings of the 28th International Conference on Developments in Language Theory. pp. 141–155. No. 14791 in LNCS, Springer, Göttingen, Germany (2024). https://doi.org/10.1007/978-3-031-66159-4_11
7. Harrison, M.A.: Introduction to Formal Language Theory. Addison-Wesley (1978)
8. Holzer, M., Rauch, C.: On Jaffe's pumping lemma, revisited. In: Bordihn, H., Tran, N., Vaszil, G. (eds.) Proceedings of the 25th International Conference on Descriptional Complexity of Formal Systems. pp. 65–78. No. 13918 in LNCS, Springer, Potsdam, Germany (2023). https://doi.org/10.1007/978-3-031-34326-1_5
9. Holzer, M., Rauch, C.: On minimal pumping constants for regular languages. In: Gazdag, Z., Iván, S., Kovásznai, G. (eds.) Proceedings of the 16th International Conference on Automata and Formal Languages. pp. 127–141. No. 386 in EPTCS, Eger, Hungary (2023). https://doi.org/10.4204/EPTCS.386.11
10. Holzer, M., Rauch, C.: The range of state complexities of languages resulting from the cascade product—the unary case. Internat. J. Found. Comput. Sci. **34**(8), 987–1022 (2023). https://doi.org/10.1142/S0129054123430049
11. Hricko, M., Jirásková, G., Szabari, A.: Union and intersection of regular languages and descriptional complexity. In: Mereghetti, C., Palano, B., Pighizzini, G., Wotschke, D. (eds.) Proceedings of the 7th Workshop on Descriptional Complexity of Formal Systems. pp. 170–181. Universita degli Studi di Milano, Como, Italy (2005)
12. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. System Sci. **63**(4), 512–530 (2001). https://doi.org/10.1006/jcss.2001.1774
13. Jaffe, J.: A necessary and sufficient pumping lemma for regular languages. SIGACT News **10**(2), 48–49 (Sommer 1978). https://doi.org/10.1145/990524.990528
14. Kozen, D.C.: Automata and Computability. Undergraduate Texts in Computer Science, Springer (1997). https://doi.org/10.1007/978-1-4612-1844-9

15. Nijholt, A.: YABBER—yet another bibliography: Pumping lemma's. An annotated bibliography of pumping. Bull. EATCS **17**, 34–53 (1982)
16. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)
17. Pighizzini, G., Shallit, J.: Unary language operations, state complexity and Jacobsthal's function. Internat. J. Found. Comput. Sci. **13**(1), 145–159 (2002). https://doi.org/10.1142/S012905410200100X
18. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM J. Res. Dev. **3**, 114–125 (1959). https://doi.org/10.1147/rd.32.0114
19. Savitch, W.J.: Relationships between nondeterministic and deterministic tape complexities. J. Comput. System Sci. **4**(2), 177–192 (1970). https://doi.org/10.1016/S0022-0000(70)80006-X
20. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time. In: Proceedings of the 5th Symposium on Theory of Computing. pp. 1–9 (1973)
21. Wagner, K.W.: Bounded query classes. SIAM J. Comput. **19**(5), 833–846 (1990). https://doi.org/10.1137/0219058