On Pumping Preserving Homomorphisms and the Complexity of the Pumping Problem (Extended Abstract)

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

[1] Planerio GmbH, Theresienhöhe 11A, 80538 München
h.gruber@planerio.de
[2] Institut für Informatik, Universität Giessen
Arndtstr. 2, 35392 Giessen, Germany
{holzer,christian.rauch}@informatik.uni-giessen.de

Abstract. This paper complements a recent inapproximability result for the minimal pumping constant w.r.t. a fixed regular pumping lemma for nondeterministic finite automata [H. Gruber and M. Holzer and C. Rauch. The Pumping Lemma for Regular Languages is Hard. *CIAA 2023*, pp. 128-140.], by showing the inapproximability of this problem even for deterministic finite automata, and at the same time proving stronger lower bounds on the attainable approximation ratio, assuming the Exponential Time Hypothesis (ETH). To that end, we describe those homomorphisms that, in a precise sense, preserve the respective pumping arguments used in two different pumping lemmata. We show that, perhaps surprisingly, this concept coincides with the classic notion of star height preserving homomorphisms as studied by McNaughton, and by Hashiguchi and Honda in the 1970s. Also, we gain a complete understanding of the minimal pumping constant for bideterministic finite automata, which may be of independent interest.

1 Introduction

The investigation of combinatorial properties and decision problems dates back to the very early days of automata and formal language theory. One of the most well known combinatorial properties that are satisfied by context-free and regular languages are pumping or interchange lemmata. For regular languages one finds, e.g., the following pumping lemma in Kozen's monograph on automata and computability—the lemma describes a necessary condition for languages to be regular [15, 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 $t \geq 0$—it is then said that y can be* pumped *in w.*

A lesser-known pumping lemma, attributed to Jaffe [14], characterizes the regular languages, by describing a necessary and sufficient condition for languages to be regular. For other pumping lemmata see, e.g., the annotated bibliography on pumping [18]:

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 $v \in \Sigma^*$.

These lemmata are part of the automata and formal language standard toolbox and their combinatorial properties are well understood. But what about their computational properties? Recently this question was answered in [8] by studying the Pumping-Problem for finite automata. This is, given a finite automaton A and a value p, does the statement of Lemma 1 (or alternatively of Lemma 2) hold for the language $L(A)$ w.r.t. the value p? It turned out that this problem is already intractable for DFAs, i.e., coNP-complete, regardless of whether we check for the pumping property described by Kozen [15] or Jaffe [14]. This is quite remarkable since it is a rare example of a finite automaton problem where already, for a single device, the studied property becomes intractable. The latter pumping property is more complex for NFAs, namely PSPACE-complete, while the former is 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 also an inapproximability result was shown for both pumping properties unless the Exponential Time Hypothesis (ETH) fails. Whether one can come up with a similar inapproximability result for DFAs was left open, and this is also the starting point of this research.

The exponential time hypothesis—roughly speaking this is an unproven computational hardness assumption which states [13] that the satisfiability of n variable 3-SAT cannot be solved in sub-exponential time $2^{o(n)}$—, as well as related assumptions which are stronger than $\mathsf{P} \neq \mathsf{NP}$, emerged as a swiss-army knife, which allows for a fine-grained analysis of NP-hard problems. To name a few, it helps to gauge the inherent limits of approximability beyond polynomial time, as well as those of parameterized and exact exponential algorithms [3]. In recent years, numerous algorithmic impossibility results based on the ETH and related hypotheses have been derived [16, 21], more recently also for problems related to finite automata and regular expressions, see, e.g., [4, 6, 9].

The main result of the present paper is an inapproximabilty bound for deterministic finite automata for both pumping properties unless ETH fails. This supersedes the previous inapproximability result for NFAs from [8]. The ingredients for this result are (i) a notion we call pumping preserving homomorphism,

[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.*

which allows a binary encoding of the problem and (ii) a non-trivial relation between the longest path problem on directed graphs and minimal pumping constants for the two pumping lemmata mentioned above. These two ingredients allow for a reduction from the inapproximability of the longest path problem on directed graphs [2] to the PUMPING-PROBLEM for binary regular languages unless $\mathsf{P} = \mathsf{NP}$. Also, assuming ETH, which is a stronger assumption than $\mathsf{P} \neq \mathsf{NP}$, it cannot be approximated in polynomial time within an even higher factor. The obtained inapproximability bounds significantly outperform the previously known bounds from [8] for NFAs. Due to space constraints, all proofs can be found in the full version of this paper.

2 Preliminaries

Next we fix some definitions on finite automata—cf. [10]. 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 Homomorphisms Preserving the Pumping Property

We investigate the impact of homomorphisms on pumping. To achieve this, we introduce the concept of homomorphisms that maintain pumping (pumping preserving homomorphisms). This concept depends on the way the pumping has to be performed—compare Lemma 1 and Lemma 2– and is closely connected to the idea of star-height preserving homomorphisms [11, 17]. The latter have proven to be highly advantageous in exploring issues related to the descriptional complexity of regular expressions—see, e.g., [7]. The property of pumping preservation exhibited by homomorphisms serves a comparable *rôle* in the examination of the computational complexity of pumping problems, as studied in [8].

In fact, our proofs below characterizing the different variants of pumping preserving homomorphisms mirroring similar properties of the star height preserving homomorphisms, culminating in a proof that all these notions coincide. For most of the examples and proofs below, our strategy follows the same outline as in [11]. We try to keep this paper self-contained, but occasionally need some technical lemmata from [11], whose proofs are not essential for understanding the material presented here.

Definition 3. *Let $L \subseteq \Sigma^*$ be a regular language and w be a word in Σ^*. Then we define the following two properties:*

1. *The word w has the pumping property w.r.t. the language L, if w admits a decomposition $w = xyz$ with $|y| \geq 1$ such that $xy^*z \subseteq L$, and*
2. *the word w has the enhanced pumping property w.r.t. the language L, if w admits a decomposition $w = xyz$ with $|y| \geq 1$ such that[4] $xyzv \subseteq L$ if and only if $xy^*zv \subseteq L$, for all words $v \in \Sigma^*$.*

Let $h : \Sigma \to \Gamma^*$ be a homomorphism. Then h is said to preserve the pumping property if and only if the following condition holds: for each regular language L over Σ and each word $w \in \Sigma^*$, the word w has the pumping property w.r.t. the language L if and only if its homomorphic image $h(w)$ has the pumping property w.r.t. $h(L)$. A similar definition applies for a homomorphism to be enhanced pumping preserving *by replacing pumping preserving by enhanced pumping preserving everywhere.*

The following is immediate from the definition of the pumping properties.

Lemma 4. *Let $L \subseteq \Sigma^*$ be a regular language and w be a word in Σ^*. If the word w* does not *satisfy the pumping property w.r.t. the language L, then w* does

[4] In abuse of notation we write $xyzv \subseteq L$ instead of $\{xyzv\} \subseteq L$.

not *satisfy the enhanced pumping property w.r.t. the same language. Moreover, if the homomorphism h* does not *preserve the pumping property, then it* does not *preserve the enhanced pumping property either.* □

Next we show that one implication in the definition of the (enhanced) pumping preserving property is easily seen to be true.

Lemma 5. *For each ε-free homomorphism $h : \Sigma \to \Gamma^*$ and every regular language L over Σ, and every word $w \in \Sigma^*$ the following holds: if w has the pumping property w.r.t. L, then $h(w)$ has the pumping property w.r.t. $h(L)$. The statement is also valid for the enhanced pumping property.*

What are the homomorphisms such that the reverse implication is satisfied as well? The notion of preserving the (enhanced) pumping property is well defined, although one at first glance might think that its always true for any (ε-free) homomorphism, since regular languages can be pumped and are closed under homomorphisms. This is not the case.[5]

Example 6. Let $\Sigma = \{a, b\}$ let $\Gamma = \{c\}$, and let h denote the unary projection $h(a) = h(b) = c$. Consider the language $L = (aa)^* \cup b(bb)^*$. Then the word $w = b$ cannot be pumped w.r.t. L, since $b^i \notin L$ whenever i is odd. But $h(L) = c^*$, and it is obvious that the word $h(b)$ can be pumped w.r.t. $h(L)$. □

In fact, all homomorphisms preserving the (enhanced) pumping property are injective, as we shall prove later on. Recall that an injective homomorphism is commonly referred to as *code*, and the homomorphic images of the alphabet letters of its domain are referred to as *codewords*. A code with the property that no codeword is a prefix (suffix, respectively) of another codeword is a *prefix code* (*suffix code*, respectively). A code that is both a prefix code and a suffix code is called a *bifix code*.

Example 7. Let $\Sigma = \{a, b, c\}$, let $\Gamma = \{0, 1\}$. Let h be the prefix code given by $h(a) = 01$, $h(b) = 011$, and $h(c) = 0111$. Consider the language $L_1 = (a \cup bc^*b)^+$. Then the word $w = bb$ *does not* have the pumping property: there are two decompositions $w = xyz$ with $|y| > 0$. But, for these, we have $xy^0z = b$ or $xy^0z = \varepsilon$, which are not in L.

Now let $L_2 = 0(10 \cup 1101)^*1$. Then obviously $0(1101)^i1 \in L_2$, for every value $i \geq 0$, so the word 011011 has the pumping property w.r.t. language L_2. Observe, that for all $i \geq 1$ we have

$$h(a^{i+1}) = (01)^{i+1} = 0\underbrace{(10)\cdots(10)}_{i \text{ times}}1$$

as well as

$$h(bc^ib) = 011\underbrace{(0111)\cdots(0111)}_{i \text{ times}}011 = 0\underbrace{(1101)\cdots(1101)}_{i+1 \text{ times}}1,$$

[5] Because of Lemma 4, in the examples to come, it suffices to restrict our attention to the (non-enhanced) pumping property.

for all $i \geq 0$. One can easily observe that these two patterns exhaust the set L_2, so we can conclude that $L_2 = h(L_1)$. □

The problematic phenomenon illustrated in the above example has been formalized in [11]: Let $h : \Sigma^* \to \Gamma^*$ be a code. Then, we say that h has the *non-crossing property*, if and only if for all $v_1, v_2, w_1, w_2 \in \Gamma^+$ we have that if $(v_1 \cup v_2) \cdot (w_1 \cup w_2) \subseteq h(\Sigma)$, then $v_1 = v_2$ or $w_1 = w_2$.

Lemma 8. *Let h be a code which* does not *have the non-crossing property. Then h is neither pumping preserving nor enhanced pumping preserving.*

If we restrict our attention to bifix codes, the non-crosssing property is sufficient:

Lemma 9. *Let h be a bifix code with the non-crossing property, and let L be a regular language. If $w \in L$* does not *have the pumping property w.r.t. language L, then $h(w)$* does not *have the pumping property w.r.t. $h(L)$. The result is also valid for the enhanced pumping property.*

Now, we are ready to prove that indeed all pumping preserving homomorphisms are codes:

Lemma 10. *Let h be a homomorphism that is* not *injective. Then h is neither pumping preserving, nor enhanced pumping preserving.*

But we observe that in general it is not needed for a pumping preserving homomorphism to be prefix-free.

Lemma 11. *There are (enhanced) pumping preserving codes which are not prefix codes.*

To obtain a necessary and sufficient condition, we use a few more definitions from [11]: Let $h : \Sigma^* \to \Gamma^*$ be a code. Let R_1, R_2 be languages over Γ such that $R_1 \cdot R_2 \subseteq h(\Sigma^*)$. The ordered pair (R_1, R_2) *has the tag* (w.r.t. h) if one of the following holds:

- There exists $r \in \Gamma^*$ and $R_1' \subseteq h(\Sigma^*)$ such that $R_1 = R_1' r$ and $rR_2 \subseteq h(\Sigma^*)$.
- There exists $s \in \Gamma^*$ and $R_2' \subseteq h(\Sigma^*)$ such that $R_2 = sR_2'$ and $R_1 s \subseteq h(\Sigma^*)$.

The shortest word that satisfies the first condition (the second condition, respectively) is called the suffix tag (the prefix tag, respectively) of (R_1, R_2). A code *has the tag property* if for all $x, x', y, y' \in \Gamma^*$, if $(x \cup x')(y \cup y') \subseteq h(\Sigma^*)$, then the pair $(x \cup x', y \cup y')$ has the tag.

For the proof of the next result, we need a theorem on unique decodability as stated in [11] and attributed to Schützenberger [20].

Theorem 12. *A homomorphism $h : \Sigma^* \to \Gamma^*$ is injective if and only if the following hold:*

- *For all $a \in \Sigma$, $h(a) \notin h(\Sigma \setminus \{a\})^*$.*

– *For all $x, y, z \in \Sigma^*$ the following holds: if all of x, xy, yz, and z are in $h(\Sigma^*)$, then $y \in h(\Sigma^*)$.*

Now we are equipped for proving the next statement—the proof of the following lemma is similar to that in [11, Theorem 5.1].

Lemma 13. *Let $h : \Sigma^* \to \Gamma^*$ be a code that has the tag property. Then h is pumping and enhanced pumping preserving.*

The converse direction is now easy:

Lemma 14. *If a code h is pumping preserving, then it has the tag property. The implication remains valid if h is enhanced pumping preserving.*

Hence, we have obtained a tight characterization of those homomorphisms that preserve the pumping properties.

Corollary 15. *A homomorphism preserves the (enhanced) pumping property if and only if it has the tag property.* $\square$

Finally, we can state the main theorem of this section. It was proved in [11] that a homomorphism preserves star height if and only if it has the tag property. Thus, we obtain:

Theorem 16. *A homomorphism preserves the pumping property if and only if it preserves the enhanced pumping property if and only if preserves star height.*

4 Minimal Pumping Constants and Longest Paths in Finite Automata

We consider the pumping constants $\mathtt{mpc}(L)$ and $\mathtt{mpe}(L)$ and their relation to the longest path in the minimal finite automaton accepting the regular language L. Here $\mathtt{mpc}(L)$ ($\mathtt{mpe}(L)$, respectively) refers to the minimal number p satisfying the conditions of Lemma 1 (Lemma 2, respectively), for a regular language L over Σ. Simple facts about these constants can be found in [5, 12]. Concerning the relation between both $\mathtt{mpc}$ and $\mathtt{mpe}$ we have $\mathtt{mpc}(L) \leq \mathtt{mpe}(L)$, for every regular language L. The relation of $\mathtt{mpe}(L)$ and the state complexities is more subtle, namely for a regular language L over the alphabet Σ we have $\mathtt{mpc}(L) \leq \mathtt{sc}(L)$ and it was shown in [12] that

$$\mathtt{mpe}(L) \leq \mathtt{sc}(L) \leq \sum_{i=0}^{\mathtt{mpe}(L)-1} |\Sigma|^i.$$

The former inequality also holds for $\mathtt{nsc}$, the nondeterministic state complexity, i.e., $\mathtt{mpc}(L) \leq \mathtt{nsc}(L)$, while the latter does not generalize. In fact, $\mathtt{mpe}$ and $\mathtt{nsc}$ are incomparable [8]. There it was argued that the $\mathtt{mpe}$-measure and the length

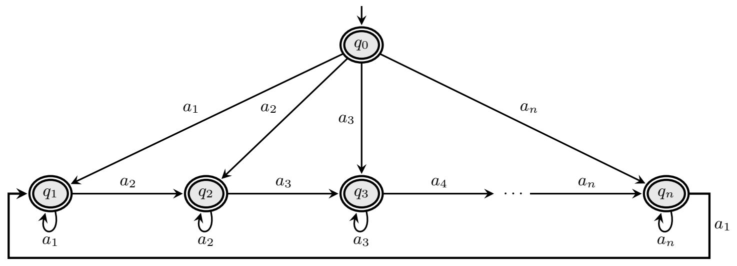

Fig. 1. The deterministic automaton A_n accepting $L_n = L(A_n)$. The non-accepting sink state is not shown. The language L_n satisfying $\texttt{mpe}(L_n) = 3$ and the longest path in A starting in the initial state q_0 is of length $n + 1$.

of the *longest path* of the automaton, that is, a simple directed path of maximum length from the initial state of the automaton, are different measures in general, using the witness shown in Figure 1. Nevertheless, for a restricted class of automata we will show that in fact both pumping measures can be bounded somehow with the length of the longest path of the underlying automaton. Before we state our results for the pumping measures and their relation to the longest path, we introduce two notations: let A be a DFA and q a state of A, not necessarily final. Then ℓ_A ($\ell_A\big|_q$, respectively) refers to the length, i.e., number of transitions, of the longest simple directed path of maximum length in the automaton A starting in q_0 and ending in any state (in state q, respectively). Here a path is called *simple* if it does not have repeated states/vertices. The following observation is immediate by Jaffe's proof, cf. [14], and was mentioned in [8]:

Lemma 17. *Let A be a DFA and $L := L(A)$. Then $\texttt{mpe}(L) \leq \ell_A + 1$. If L is a unary language, then $\texttt{mpe}(L) = \texttt{sc}(L)$.*

Using the witness depicted in Figure 1, we have a language L where $\texttt{mpe}(L)$ and $\ell_A + 1$ can be far apart. Nevertheless, for the class of bideterministic finite automata (biDFAs, for short) this is not the case as shown next. Here, a finite automaton A is *bideterministic* if it is both partially deterministic and partially co-deterministic and has a sole accepting state. Moreover, an automaton A is *partially co-deterministic* if the reversed automaton obtained by reversing the transitions of A is partially deterministic. A language L is said to be *bideterministic* if it is accepted by a biDFA A, i.e., $L = L(A)$. Observe that a language L is accepted by a bideterministic finite automaton if and only if the minimal automaton of L is reversible, i.e., deterministic and co-deterministic, and has a unique final state [19]. This leads us to the key property of bideterministic finite automata of which we will make heavy use:

Lemma 18. *Let $A = (Q, \Sigma, \cdot, q_0, F)$ be a biDFA and z a word from Σ^*. Then $p \cdot z = q$, for $p, q \in Q$ implies that $|\{ p \mid p \cdot z = q \}| = 1$, i.e., if the z-predecessor state p of q exists, then p is unique.* □

The next lemma uses the above property in the proof and reads as follows:

Lemma 19. *Let L be a bideterministic language accepted by a minimal DFA A and assume $L := L(A)$. Then $\ell_A \leq \mathtt{mpe}(L) \leq \ell_A + 1$.*

The bounds in the previous lemma are best possible. This can be seen by the following bideterministic languages: language $L_1 = \{ab\}$ over the binary alphabet $\{a, b\}$ and the unary language $L_2 = \{aa\}$. Both languages are accepted by minimal 4-state DFAs, which are both bideterministic if the non-accepting sink state is removed. Let A_1 (A_2, respectively) be the DFA that accepts L_1 (L_2, respectively)—see Figure 2. Thus, in both cases we have $\ell_{A_1} = \ell_{A_2} = 3$,

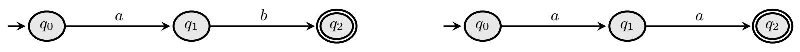

Fig. 2. Deterministic finite automata A_1 and A_2 accepting the bideterministic and finite languages $L_1 = \{ab\}$ and $L_2 = \{aa\}$, respectively. In both drawings the non-accepting sink state is not shown. The longest simple path in both automata is of length 3 (including the non-accepting sink state) and the pumping constants of the languages are $\mathtt{mpe}(L_1) = 3$ and $\mathtt{mpe}(L_2) = 4$.

but $\mathtt{mpe}(L_1) = 3 = \ell_{A_1}$, while $\mathtt{mpe}(L_2) = 4 = \ell_{A_2} + 1$. This is seen as follows: first consider the language L_1. The word ab cannot be pumped at all, because any shorter word is not a member of L_1. Hence, $\mathtt{mpe}(L_1) \geq 3$. Every word of length at least 3 maps the initial state of the automaton A_1 to the non-accepting sink state. Any word of length at least 3 that starts with the letter b or with the prefix aa visits the non-accepting sink state at least twice and hence can by pumped respecting equivalence classes. In case the word of length at least 3 begins with ab we are left with the prefixes aba or abb. Then, it is easy to see that in the former case the letter b in the middle of the prefix can be pumped, while in the latter case the first letter a is pumpable. The details are left to the reader. Hence, we have $\mathtt{mpe}(L) = 3 = \ell_{A_1}$ as required. For the unary language L_2, the argumentation that $\mathtt{mpe}(L_2) = 4$ is slightly simpler. Since L_2 is unary, there is only a single word of length 3, but a^3 is not a member of L_2. Shortening this word to any length strictly less than 3, results in a word that belongs to a different Myhill-Nerode equivalence class then the word a^3. Hence a^3 cannot be pumped by respecting equivalence classes. Therefore, $\mathtt{mpe}(L_2) \geq 3+1 = 4$ and by Lemma 17 we have $\mathtt{mpe}(L) \leq \ell_{A_2} + 1 = 3 + 1 = 4$. Thus, $\mathtt{mpe}(L_2) = 4 = \ell_{A_2} + 1$.

Next, let us consider the pumping measure $\mathtt{mpc}$. Recall that $\mathtt{mpc}(L) \leq \mathtt{mpe}(L)$, for every regular language L. The gap between path measures can be arbitrarily large even for bideterministic languages. Consider the bideterministic unary language $L_n = (a^n)^*$, for $n \geq 1$, which is accepted by a unary cyclic DFA with n states. Since $\mathtt{mpe}$ and $\mathtt{sc}$ coincides for unary languages, we have that $\mathtt{mpe}(L_n) = n$. On the other hand, $\mathtt{mpc}(L_n) = 1$, because $L_n \neq \emptyset$ and each word $w \in L_n$ can be pumped by choosing $x = \lambda$, $y = w$, and $z = \lambda$, since

$xy^*z = w^* \subseteq L$. Nevertheless, we can also show a nice relation for the mpc-measure with a longest path problem. The result is stated in the following lemma.

Lemma 20. *Let L be a bideterministic language that is accepted by the minimal DFA A and define $L := L(A)$. Then $\mathtt{mpc}(L) = \ell_A\big|_{q_f} + 1$, where q_f is the unique final state of A.*

5 The Complexity of Pumping, Revisited

We will consider the following decision problem [8] 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 1 holds for the language $L(A)$ w.r.t. the value p.

We apply our findings on the relation between the minimal pumping constants from the previous section in order to give a simpler proof for the coNP-completeness of the Pumping-Problem for DFAs. This will be a corollary to an inapproximability result for DFAs, which significantly improves a previously known inapproximability result for NFAs from [8].

The Longest-Directed-Path-Problem (LDP-problem) is defined as follows: given a directed graph $G = (V, E)$, find the longest sequence of distinct vertices $v_1, v_2, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$, for $1 \leq i < k$. Naturally, the LDP-problem gives rise to a NP-complete decision and an approximation problem. The LDP-optimization problem cannot be approximated in polynomial time within a factor of $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$, unless $\mathsf{P} = \mathsf{NP}$. Also, assuming the Exponential Time Hypothesis (ETH), which is a stronger assumption than $\mathsf{P} \neq \mathsf{NP}$, it cannot be approximated in polynomial time within a factor of $\omega(\frac{n \log\log n}{(\log n)^2})$. Both inapproximability bounds are shown[6] in [2]. By inspecting the proof, one can see that the result in fact applies to digraphs with bounded outdegree [1]. In the next theorem we use the LDP-optimization problem as a starting point for our reduction to the Pumping-Problem for DFAs.

Theorem 21. *Let A be a DFA with s states over an alphabet of size $O(s)$. Then no deterministic polynomial time algorithm can approximate the minimal pumping constant w.r.t. Lemma 1 (Lemma 2, respectively) within a factor of $s^{1-\varepsilon}$ for any constant $\varepsilon > 0$, unless $\mathsf{P} = \mathsf{NP}$. Assuming ETH, the inapproximability factor becomes $\omega(\frac{s \log\log s}{(\log s)^2})$.*

A direct consequence of the previous proof is that the Pumping-Problem for both pumping lemmata is intractable; this re-establishes a result from [8]—with

[6] The actual inapproximability bound assuming the ETH from [2] is slightly stronger, but involves a function $f(n)$ satisfying some additional tameness constraint. We use the simplified bound for better readability.

an even stronger bound on the approximation ratio—, but it uses a growing-size alphabet. Luckily, we can use a (enhanced) pumping preserving homomorphism to code it down to a binary alphabet. Let $\Sigma = \{a_1, a_2, \ldots, a_r\}$ be an r-letter alphabet and h be the homomorphism given by $a_i \mapsto \text{bin}(i)\text{bin}(i)^R$, for $1 \leq i \leq r$. Here, $\text{bin}(i)$ denotes the $\lceil \log r \rceil$-bit binary encoding of the number i and $\text{bin}(i)^R$ its reversal. As shown in [7], the code h preserves star height, which in turn implies that h also preserves the (enhanced) pumping property by Theorem 16. Now we are ready for the next lemma.

Lemma 22. *Let $r \geq 3$ be an integer, $\Sigma = \{a_1, a_2, \ldots, a_r\}$ be an r-letter alphabet, h be the homomorphism given by $a_i \mapsto bin(i)bin(i)^R$, for $1 \leq i \leq r$, and L be a regular language over Σ. Then*

$$\mathit{mpc}(h(L)) = 2b \cdot (\mathit{mpc}(L) - 1) + 1,$$

where $b = \lceil \log r \rceil$.

For the minimal pumping constants w.r.t. Lemma 2 we prove the following lemma in similar vein as Lemma 22.

Lemma 23. *Let $r \geq 3$ be an integer, $\Sigma = \{a_1, a_2, \ldots, a_r\}$ be an r-letter alphabet, h be the homomorphism given by $a_i \mapsto bin(i)bin(i)^R$, for $1 \leq i \leq r$, and L be a regular language over Σ. Then*

$$2b \cdot (\mathit{mpe}(L) - 1) < \mathit{mpe}(h(L)) \leq 2b \cdot \mathit{mpe}(L),$$

where $b = \lceil \log r \rceil$.

One may ask whether we can come up with an exact calculation of $\mathtt{mpe}(h(L))$ like for $\mathtt{mpc}(h(L))$. As we will see next this is not possible since the bounds for $\mathtt{mpe}(h(L))$ from the previous lemma are best possible. We begin with a language L that meets the upper bound, i.e., $\mathtt{mpe}(h(L)) = 2b \cdot \mathtt{mpe}(L)$. Define $L = (abc)^*$. Thus, $2b = 4$. Observe that each word of length at least three is pumpable w.r.t. Lemma 2 either (i) by its prefix abc, (ii) by its second letter if the first letter is not an a, (iii) by its first letter if the second letter is not an a, or (iv) by its third letter, otherwise. Thus $\mathtt{mpe}(L) \geq 3$. Moreover, the word ab cannot be pumped w.r.t. Lemma 2, which can easily be seen by concatenating it with $v = c$. Hence, we have $\mathtt{mpe}(L) = 3$. Next, consider $h(L)$. The language L is mapped by h onto $h(L) = (000001101001)^*$. Similarly, as in the above argumentation, one finds that each word in $h(L)$ of length twelve can be pumped w.r.t. Lemma 2 while the word 00000110100 cannot be pumped. Therefore, $\mathtt{mpe}(h(L)) = 12 = 4 \cdot 3 = 2b \cdot \mathtt{mpe}(L)$ as desired. Finally, we consider a language L that meets the lower bound, that is, $\mathtt{mpe}(h(L)) = 2b \cdot (\mathtt{mpe}(L) - 1) + 1$. Let $L = abc$. Again $2b = 4$. It is easy to see that $\mathtt{mpe}(L) = 4$, since the word abc cannot be pumped w.r.t. Lemma 2 while all words of length four are pumpable. The details are left to the reader. On the other hand, we have $h(L) = 000001101001$, which fulfills $\mathtt{mpe}(h(L)) = 13 > 12 + 1 = 4 \cdot 3 + 1 = 4 \cdot (4 - 1) + 1 = 2b \cdot (\mathtt{mpe}(L) - 1) + 1$. Thus, the bounds proven in Lemma 23 are best possible.

We close this section with the main result on the inapproximability of the Pumping-Problem for DFAs on binary alphabets.

Theorem 24. *Let A be a DFA with s states over a binary alphabet. Then no deterministic polynomial time algorithm can approximate the minimal pumping constant w.r.t. Lemma 1 within a factor of $s^{1-\varepsilon}$ for any constant $\varepsilon > 0$, unless* $\mathsf{P} = \mathsf{NP}$. *In case of ETH the inapproximability factor becomes $\frac{s \log\log s}{(\log s)^2}$. Both statements hold true if one considers the approximation of the minimal pumping constant w.r.t. Lemma 2.*

6 Conclusion

In the present paper, we revisited the pumping problem for regular languages. For bideterministic finite automata, the pumping constant is more well-behaved than in the general case. We characterized the pumping constant in terms of the longest path from the start state to an accepting state. Then, we turned our attention to the homomorphisms preserving the pumping property. This is seemingly a more primitive abstraction than that of a homomorphism preserving star height. Interestingly, the two definitions are equivalent. Compared with the results obtained for NFAs from [8], the inapproximability bound we obtained by putting the pieces together is stronger, even though we restricted the input to DFAs, which means that the input is represented less succinctly.

Acknowledgement. The authors would like to thank Andreas Björklund for some fruitful discussion.

References

1. Björklund, A.: Personal communication (2024)
2. Björklund, A., Husfeldt, T., Khanna, S.: Approximating longest directed paths and cycles. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) Proceedings of the 31st International Colloquim Automata, Languages and Programming. pp. 222–233. No. 3142 in LNCS, Springer, Turku, Finland (2004). `https://doi.org/10.1007/978-3-540-27836-8_21`
3. 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`
4. Czerwinski, W., Debski, M., Gogasz, T., Hoi, G., Jain, S., Skrzypczak, M., Stephan, F., Tan, C.: Languages given by finite automata over the unary alphabet. In: Bouyer, P., Srinivasan, S. (eds.) Proceedings of the 43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023). LIPIcs, vol. 284, pp. 22:1–22:20. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, IIIT Hyderabad, Telangana, India (2023). `https://doi.org/10.4230/LIPIcs.STACS.2008.1354`
5. Dassow, J., Jecker, I.: Operational complexity and pumping lemmas. Acta Inform. **59**, 337-–355 (2022). `https://doi.org/10.1007/s00236-022-00431-3`
6. Fernau, H., Krebs, A.: Problems on finite automata and the exponential time hypothesis. Algorithms **10**(1), 24 (2017). `https://doi.org/10.3390/a10010024`

7. Gruber, H., Holzer, M.: Tight bounds on the descriptional complexity of regular expressions. In: Diekert, V., Nowotka, D. (eds.) Proceedings of the 13th International Conference Developments in Language Theory. pp. 276–287. No. 5583 in LNCS, Springer, Stuttgart, Germany (2009). https://doi.org/10.1007/978-3-642-02737-6_22
8. 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
9. Gruber, H., Holzer, M., Wolfsteiner, S.: On minimizing regular expressions without kleene star. In: Bampis, E., Pagourtzis, A. (eds.) Proceedings of the 23rd International Symposium on Fundamentals of Computation Theory. pp. 245–258. No. 12867 in LNCS, Springer, Athens, Greece (2021). https://doi.org/10.1007/978-3-030-86593-1_17
10. Harrison, M.A.: Introduction to Formal Language Theory. Addison-Wesley (1978)
11. Hashiguchi, K., Honda, N.: Homomorphisms that preserve star height. Inform. Comput. **30**(3), 247–266 (1976). https://doi.org/10.1016/S0019-9958(76)90511-8
12. 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
13. Impagliazzo, R., Paturi, R.: Complexity of k-SAT. In: Sirakov, B., de Souza, P.N., Viana, M. (eds.) Proceedings of the 14th Annual IEEE Conference on Computational Complexity. pp. 237–240. IEEE Computer Society, Atlanta, Georgia, USA (1999). https://doi.org/10.1109/CCC.1999.766282
14. 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
15. Kozen, D.C.: Automata and Computability. Undergraduate Texts in Computer Science, Springer (1997). https://doi.org/10.1007/978-1-4612-1844-9
16. Lakshtanov, D., Marx, D., Saurabh, S.: Lower bounds based on the exponential time hypothesis. Bulletin of the European Association for Theoretical Computer Science **105**, 41–72 (2011)
17. McNaughton, R.: The loop complexity of regular events. Information Sciences **1**, 305–328 (1969). https://doi.org/10.1016/S0020-0255(69)80016-2
18. Nijholt, A.: YABBER—yet another bibliography: Pumping lemma's. An annotated bibliography of pumping. Bull. EATCS **17**, 34–53 (1982)
19. Pin, J.E.: On reversible automata. In: Proceedings of 1st Latin American Symposium on Theoretical Informatics. pp. 401–416. No. 583 in LNCS, Springer, São Paulo, Brazil (1992). https://doi.org/10.1007/BFb0023844
20. Schützenberger, M.P.: Une théorie algébrique du codage. Séminaire Dubreil. Algèbre et théorie des nombres **9**, 1–24 (1955-1956)
21. Williams, V.V.: On some fine-grained questions in algorithms and complexity. In: Sirakov, B., de Souza, P.N., Viana, M. (eds.) Proceedings of the International Congress of Mathematicians. pp. 3447–3487. World Scientific, Rio de Janeiro, Brazil (2019). https://doi.org/10.1142/9789813272880_0188