I F I G
Research
Report

Institut für Informatik

On Pumping Preserving Homomorphisms and the Complexity of the Pumping Problem

Hermann Gruber Markus Holzer Christian Rauch

IFIG Research Report 2401
June 2024

Institut für Informatik
JLU Gießen
Arndtstraße 2
35392 Giessen, Germany
Tel: +49-641-99-32141
Fax: +49-641-99-32149
mail@informatik.uni-giessen.de
www.informatik.uni-giessen.de

IFIG Research Report
IFIG Research Report 2401, June 2024

On Pumping Preserving Homomorphisms and the Complexity of the Pumping Problem

Hermann Gruber[1]
Planerio GmbH, Theresienhöhe 11A, 80538 München
and
Markus Holzer[2] and Christian Rauch[3]
Institut für Informatik, Universität Giessen
Arndtstr. 2, 35392 Giessen, Germany

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.

MSC Classification:
68Q05: Models of computation (Turing machines, etc.)
68Q17: Computational difficulty of problems (lower bounds, completeness, difficulty of approximation, etc.)
68Q45: Formal languages and automata

Additional Key Words and Phrases: Finite Automata, Pumping Lemmata, Star Height Preserving Homomorphism, Pumping Preserving Homomorphism, Pumping Problem, Decision Problem, Computational Complexity, Approximiation, Exponential Time Hypothesis.

[1] Email: h.gruber@planerio.de
[2] Email: holzer@informatik.uni-giessen.de (Corresponding Author)
[3] Email: christian.rauch@informatik.uni-giessen.de

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*[1]

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

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].

[1] 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.*

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, 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.

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[2] $xyzv \subseteq L$ 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 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.*

Proof. The statement for the pumping property is immediate from the definition. For the enhanced pumping property we have to argue as follows: by definition $h(w)$ has the enhanced pumping property if and only if there is decomposition $h(w) = xyz$ with $|y| \geq 1$ such that $xyzv \subseteq h(L)$ iff $xy^*zv \subseteq h(L)$ for all words $v \in \Sigma^*$. We observe that for all words $v \in h(\Sigma^*)$ the statement of the lemma follows. If $v \in \Sigma^* \setminus h(\Sigma^*)$ then we observe that for all words w we have $h(w)v \notin h(L)$.

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

On the other hand, if we have a decomposition of $h(w)$ like mentioned before and if there is a word $v \in h(\Sigma^*)$ such that $xy^*zv \subseteq h(L)$ then we directly obtain that xy^*z is a codeword.[3] Hence, we have for all words $v \in \Sigma^* \setminus h(\Sigma^*)$ that xy^*zv is not a subset of $h(L)$, because it is not even in $h(\Sigma^*)$. Therefore, $h(w) = xyz$ is an enhanced pumpable decomposition for all words $v \in \Sigma^*$. □

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.[4]

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

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

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.*

[3] If there is no such word, we exchange L by its complement $\overline{L}$ in the whole proof.

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

Proof. By Lemma 4 it suffices to show that h is *not* pumping preserving. Since $h : \Sigma \to \Gamma^*$ does not have the non-crossing property, there are nonempty words $v_1, v_2, w_1, w_2 \in \Gamma^+$ with $v_1 \neq v_2$ and $w_1 \neq w_2$ such that the inclusion $(v_1 \cup v_2)(w_1 \cup w_2) \subseteq h(\Sigma)$ holds. Then $v_1w_1 \neq v_2w_2$ or $v_2w_1 \neq v_1w_2$, as is easily observed—see also [11, Lemma 4.4]. Note that we can exchange the *rôles* of v_1, v_2, w_1, w_2 if necessary such that $v_1w_1 \neq v_2w_2$ holds, so let us assume that this is the case. Let us denote $a = h^{-1}(v_1w_1)$, $b = h^{-1}(v_1w_2)$, $c = h^{-1}(v_2w_1)$, and $d = h^{-1}(v_2w_2)$, while keeping in mind that some of the variables a, b, c, d may, or may not, denote the same alphabet letter from Σ—we shall inspect this later.

Now consider the language $h(L) = v_1(w_1v_1 \cup w_2v_2)^*w_1$. The word $v_1w_2v_2w_1$ can obviously pumped w. r. t. the language $h(L)$. Next, it can be readily verified that $h\left((a \cup bd^*c)^+\right) = h(L)$. Since h is a code, we can deduce the equality $L = (a \cup bd^*c)^+$. But $h^{-1}(v_1w_2v_2w_1) = bc$ can have the pumping property w. r. t. language L only if $a = b$ or $a = c$: The word a is the only word in L that is shorter than bc. Hence, for pumping bc, we need to find a decomposition of bc into subwords $bc = xyz$ such that $|y| > 0$ and $xy^0z = a$.

Next, it is immediate from $w_1 \neq w_2$ that $v_1w_1 \neq v_1w_2$. Then by injectivity of h, we have $h^{-1}(v_1w_1) \neq h^{-1}(v_1w_2)$, that is, $a \neq b$. Similarly, it is immediate from $v_1 \neq v_2$ that we get $v_1w_1 \neq v_2w_1$. Then again by injectivity of h, we have $h^{-1}(v_1w_1) \neq h^{-1}(v_2w_1)$, that is, $a \neq c$. Thus bc does not have the pumping property w. r. t. L, while $h(bc)$ has the pumping property w. r. t. $h(L)$, and thus h is not 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.*

Proof. We first prove the statement for the pumping property and explain later what has to be changed for the enhanced pumping property. For the sake of contradiction, assume that w cannot be pumped w. r. t. L, but its homomorphic image $h(w)$ has the pumping property w. r. t. the language $h(L)$. That is, there is a decomposition $h(w) = xyz$ with $|y| > 0$ such that $xy^*z \subseteq h(L)$.

Here, any of x, y, z may, or may not, be in $h(\Sigma^*)$. To further investigate the possible cases, let us further decompose $x = x_1x_2$, $y = y_1y_2y_3$ and $z = z_1z_2$, with x_1 being the longest prefix of x which is a codeword, with z_2 being the longest suffix of z which is a codeword, with $x_2y_1 \in h(\Sigma)$ if x_2 is nonempty and $x_2y_1 = \varepsilon$ otherwise, and finally with $y_3z_1 \in h(\Sigma)$ if z_1 is nonempty and $y_3z_1 = \varepsilon$ otherwise. Observe, that there is a unique such decomposition $(x_1, x_2, y_1, y_2, y_3, z_1, z_2)$. Furthermore, observe, that y_2 is a codeword. First consider the case $x_2 = \varepsilon$. Pumping yields $xy^0z = x_1z_1z_2$, which is by assumption a word in $h(L)$ and thus a codeword. Since h is prefix-free and suffix-free, and z_1 cannot be a codeword by the definition of z_2, the word z_1 must be empty. But then all of x, y, and z are codewords. Then for each $i \geq 0$, the pumped image xy^iz can be decoded to a word $h^{-1}(x)(h^{-1}(y))^ih^{-1}(z)$ in $h^{-1}(h(L)) = L$, which gives the desired contradiction in this case.

The argument for the case $z_1 = \varepsilon$ is similar. So let us assume that both $x_2, z_1 \neq \varepsilon$. Since $x_2y_1 \in h(\Sigma)$, but x_2 is not in $h(\Sigma)$ by the definition of x_1, we conclude that y_1 cannot be the empty word. A similar reasoning yields that y_3 can neither be the empty word.

Now consider the pumped words $xy^2z = x_1x_2y_1y_2y_3y_1y_2y_3z_1z_2$ and $xy^0z = x_1x_2z_1z_2$. By assumption, both of these words are in $h(L)$, so they are codewords. Since h is prefix-free and suffix-free, and both x_1 and z_2 are codewords, we see that both $x_2y_1y_2y_3y_1y_2y_3z_1$ and x_2z_1 are again codewords. For the word pairs (x_2, z_1) and $(x_2y_1y_2y_3, y_1y_2y_3z_1)$, notice that not only $x_2 \cdot z_1$ and $x_2y_1y_2y_3 \cdot y_1y_2y_3z_1$ are codewords, but also their cross-wise concatenation $x_2 \cdot y_1y_2y_3z_1$

and $x_2y_1y_2y_3 \cdot z_1$, since by the definition of the decomposition $(x_1, x_2, y_1, y_2, y_3, z_1, z_2)$ we have that $x_2y_1, y_3z_1 \in h(\Sigma)$, and $y_2 \in h(\Sigma^*)$. But the existence of two word pairs (v_1, w_1) and (v_2, w_2) with $v_1 \neq v_2$ and $w_1 \neq w_2$ for which both the respective concatenations v_1w_1, v_2w_2 and their cross-wise concatenations v_1w_2, v_2w_1 are code words violates the non-crossing property of h, contradiction.

When considering the enhanced pumping property we have to take care of the following issue: let L be a regular language and let $w \in \Sigma^*$ be a word that does not have the enhanced pumped property w. r. t. L. If there is no word $v \in \Sigma^*$ such that $wv \in L$ then we exchange the language L by its complement $\overline{L}$ which does not alter argumentation because each word in Σ^* has the enhanced pumping property w. r. t. L if and only if it has the enhanced pumping property w. r. t. $\overline{L}$. Hence, we safely assume that there is a word $v \in \Sigma^*$ such that $wv \in L$. In addition these words are mapped by h into $h(\Sigma^*)$ which makes all words $h(v)$ codewords. If any word $h(w)$ has the enhanced pumping property then we know that there is a decomposition $h(w) = xyz$ with $|y| > 0$ such that

$$xyzv \subseteq h(L) \quad \text{iff} \quad xy^*zv \subseteq h(L),$$

for all words $v \in \Sigma^*$ which includes all words $v \in h(\Sigma^*)$. Thus, we restrict the reasoning to such a decomposition and all words $v \in h(\Sigma^*)$ to obtain the stated result also for the enhanced pumping property. □

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

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

Proof. Let $h : \Sigma \to \Gamma^*$ be a homomorphism which is not injective. Consider first the case that $h(a) = \varepsilon$ for some $a \in \Sigma$. Now consider the language a^*. Then the word a has the pumping property w. r. t. L. But $h(a) = \varepsilon$, and the empty word cannot be decomposed into subwords xyz with $|y| > 0$.

Now let us assume that h is ε-free and not injective; that is, there are words $w_1 \neq w_2$ such that $h(w_1) = h(w_2)$. Let $g : \{a_1, a_2\} \to \Sigma^*$ denote the homomorphism with $g(a_1) = w_1$ and $g(a_2) = w_2$. Then g must be prefix-free and suffix-free, since h is a homomorphism, $h(w_1) = h(w_2)$ and $h(a) \neq \varepsilon$ for all $a \in \Sigma$. This in turn implies unique decodability, so g is a code. It has been shown in [11, Lemma 4.6] that those codes, whose domain alphabet has size at most two, necessarily have the non-crossing property. Thus, g preserves pumping with the aid of Lemma 9. Now let $L = a_1^*a_2 \cup a_2a_1$. The word a_2a_1 does not have the pumping property w. r. t. L, since the only shorter word in L is a_2 and $a_2 = a_2a_1^0$ but $a_2a_1^2$ is not in L. But $h(g(a_2a_1)) = h(w_1)^2$ has the pumping property w. r. t. $h(g(L))$, because $h(L) = h(w_1)^{\geq 1}$ and the decomposition $h(w_1)^2 = xyz$ with $x = \varepsilon$ and $y = z = h(w_1)$ can be pumped w. r. t. $h(g(L))$. Since g preserves pumping but the function composition $h \circ g$ does not, this implies that h is not pumping preserving. By Lemma 4 it also follows that h is not enhanced pumping preserving, as desired. □

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.*

Proof. We focus on the proof for the pumping preserving property. With similar arguments as used in the proof of Lemma 9 one can show that the statement also holds for the enhanced pumping preserving property. The details are left to the reader.

Let $h : \Sigma \to \Gamma^*$ be a code with $a_1 \neq a_2 \in \Sigma$ such that $r, s \in \Gamma^+$ with $h(a_1) = r$ and $h(a_2) = rs$. Assume that h is not pumping preserving. Then there is a language L and a word $w \in L$ such that $h(w) \in h(L)$ is pumpable w. r. t. $h(L)$ but w is not pumpable w. r. t. L. Therefore we have $x, y, z \in \Gamma^*$ such that $h(w) = xyz$, $y \neq \varepsilon$, and $xy^iz \in h(L)$ for all non-negative integers i. We observe that $w \in h(L) \subseteq \{r, rs\}^*$ implies that w does not contain the subword ss. Therefore this also holds for all subwords of w, i. e., x, y, and z do not contain the ss subword.

We will find that we can derive a decomposition of w. For this we distinguish for each of the words x, y, and z whether they are in $h(\Sigma^*)$ or not:

1. For the case $x, y \in h(\Sigma^*)$ we focus on the sub-case that $z \notin h(\Sigma^*)$, because we directly obtain a decomposition of the word w by $w = h^{-1}(x)h^{-1}(y)h^{-1}(z)$, if $z \in h(\Sigma^*)$. We obtain that $x = x'r$, $y = ry'r$, and $z = sz'$, because xz, xyz, and $xyyz$ are in $h(L)$. Therefore we have $w = h^{-1}(x')a_1h^{-1}(y')a_2h^{-1}(z')$ and we can pump w by the subword $a_1h^{-1}(y')$. This can be seen as follows. It is easy to confirm that $h(h^{-1}(x')a_2h^{-1}(z')) = xz$ and for all non-negative integers $i \geq 1$ we have
$$h(h^{-1}(x')(a_1h^{-1}(y'))^ia_2h^{-1}(z')) = xy^iz.$$
Since $xy^iz \in h(L)$ for all non-negative integers $i \geq 0$ we obtain the inclusion $h^{-1}(xy^iz) \subseteq L$. Hence we found that the word w is pumpable by its subword $a_1h^{-1}(y')$.
2. For $x, z \in h(\Sigma^*)$ and $y \notin h(\Sigma^*)$ we see that $x = x'r$, $y = sy'r$, and $z = rz'$ because xz, xyz, and $xyyz$ are in $h(L)$. Hence, we have
$$w = h^{-1}(x')a_2h^{-1}(y')a_1a_1h^{-1}(z')$$
and we can pump w by the subword $a_2h^{-1}(y')$, which can be seen in similar vein as in the previous case and by observing that
$$xy^iz = h(h^{-1}(x')(a_2h^{-1}(y'))^ia_1a_1h^{-1}(z')),$$
for all non-negative integers i.
3. If $x \in h(\Sigma^*)$ and both $y, z \notin h(\Sigma^*)$ we observe that $x = x'r$, $y = sy'r$, and $z = sz'$ because xz, xyz, and $xyyz$ are in $h(L)$. Additionally $w = h^{-1}(x')a_2h^{-1}(y')a_2h^{-1}(z')$ can be pumped by its subword $a_2h^{-1}(y')$.
4. Case $x \notin h(\Sigma^*)$. We observe that $h(w) \in \{r, rs\}^* = h(L)$ implies that x is a subword of a word in $\{r, rs\}^+$. Therefore x ends either in an r or an s. Then we have $x = x'r$ or $x = x'rs$ such that $h^{-1}(x') \neq \emptyset$, which implies that $h^{-1}(x) = h^{-1}(x') \cdot \{a_1\}$ in the former case and $h^{-1}(x) = h^{-1}(x') \cdot \{a_2\}$ in the latter case.

This proves the stated claim. □

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 $R_1s \subseteq h(\Sigma^*)$.

The shortest word that satisfies the first condition (resp. the second condition) 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.*

Proof. Let L be any regular language. We will show that for each word $w \in L$, the word w can be pumped with respect to L if and only if $h(w)$ can be pumped with respect to $h(L)$. The 'only if' part of course is implied by Lemma 5, since h is a code.

The implication in the other direction requires more work. Let $h(w)$ be a word that can be pumped with respect to $h(L)$. Then there is a decomposition $h(w) = xyz$ with $|y| \geq 1$ such that $xy^*z \in h(L)$. By [11, Lemma 5.2], if a homomorphism h has the tag property, it has the tag for all pairs (R_1, R_2) with $R_1, R_2 \in \Gamma^*$ and $R_1 \cdot R_2 \subseteq h(\Sigma^*)$. Since xy^*y^*z is in $h(\Sigma^*)$, the pair (xy^*, y^*z) has the tag. Without loss of generality, we may assume it has the suffix tag r, the case of the prefix tag being symmetric. Since r is the suffix tag, every word in xy^* must have the suffix r. Let $x = x_1r$ and $y = y_1r$. Then by the tag property of the pair (xy^*, y^*z), we have

$$\begin{aligned} x_1r \in xy^* &\text{ and thus } x_1 \text{ is a codeword,} \\ x_1ry_1r \in xy^* &\text{ and thus } x_1ry_1 \text{ is a codeword,} \\ y_1rz \in y^*z &\text{ and thus } r \cdot y_1rz \text{ is a codeword,} \end{aligned}$$

and

$$z \in y^*z \text{ and thus } r \cdot z \text{ is a codeword.}$$

Now since all four words x_1, x_1ry_1rz, r_1yrz, and rz are codewords, we can apply Theorem 12 to deduce that ry_1 must be a codeword as well.

To get ready for the final *coup de grâce*, let us transform the expression xy^*z. By iterated application of the associative law for concatenation, we obtain

$$r(y_1r)^i = (ry_1)^ir \text{ for all integer } i \geq 0,$$

and thus $r(y_1r)^* = (ry_1)^*r$. We thus can deduce that for all $i \geq 0$, we have $x_1 \cdot (ry_1)^i \cdot rz = xy^iz$. With $xy^iz \in h(L)$ for all $i \geq 0$, we conclude that the preimage

$$h^{-1}(xy^iz) = h^{-1}(x_1) \cdot h^{-1}(ry_1)^i \cdot h^{-1}(rz)$$

is in L for all $i \geq 0$. Thus, the three subwords $h^{-1}(x_1)$, $h^{-1}(ry_1)$ and $h^{-1}(rz)$ form a decomposition of $h^{-1}(h(w)) = w$ that can be pumped, as desired.

With similar arguments as in the proof of Lemma 9 also the statement for the enhanced pumping property follows. □

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.*

Proof. Assume that h is a code that does not have the tag property. In [11, Lemma 5.3], it is shown that then there is another code g, such that g is prefix-free, suffix-free and has the non-crossing property, and moreover, the composition of g and h does not have the non-crossing property.

The homomorphism g is pumping preserving by Lemma 9. In contrast, with Lemma 8 it follows that $g \circ h$ is not pumping preserving. Since the composition of two pumping preserving homomorphisms is again pumping preserving, we deduce that h cannot be pumping preserving, which in turn implies that h cannot be enhanced pumping preserving either by Lemma 4. □

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.* □

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

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

Let $\mathtt{nsc}$ refer to the nondeterministic state complexity. The former inequality also holds for $\mathtt{nsc}$, 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 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, resp). 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 $\mathtt{mpe}(L) \leq \ell_A + 1$. If L is a unary language, then $\mathtt{mpe}(L) = \mathtt{sc}(L)$.*

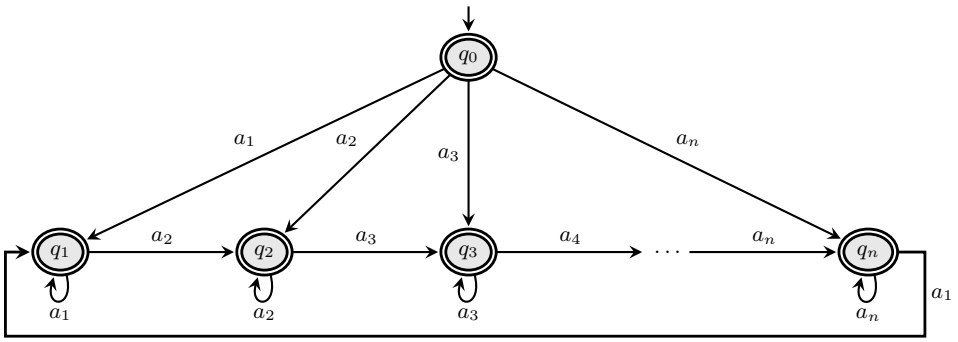

Fig. 1. The deterministic automaton A_n accepting $L_n = L(A_n)$. The non-accepting sink state is not shown. The language L_n satisfying $\mathtt{mpe}(L_n) = 3$ and the longest path in A starting in the initial state q_0 is of length $n + 1$.

Using the witness depicted in Figure 1, we have a language L where $\mathtt{mpe}(L)$ 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 L is accepted by a bideterministic finite automaton if and only if the minimal automaton of the language 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 states $p, q \in Q$, 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. Then*

$$\ell_A \leq \mathtt{mpe}(L) \leq \ell_A + 1$$

with $L := L(A)$.

Proof. First observe, that A is an ordinary DFA with a possible non-accepting sink state. Since it is minimal, A can be made bideterministic by removing the non-accepting sink state.

The upper bound is immediate by Lemma 17. For the lower bound we argue as follows: consider a longest path p in A. This path induces a word w. Depending on the state were p ends we distinguish two cases—we assume that the transition function of A is $\cdot$ and q_0 is the initial state of A:

1. Assume that the last state in the longest path p is *not* the non-accepting sink. Let q be this state. Then we consider the word w induced by the path p. Any decomposition $w = xyz$ with $|y| > 0$ gives rise to three states $q_x = q_0 \cdot x$, $q_{xy} = q_x \cdot y$, and $q_{xyz} = q_{xy} \cdot z$, where all states are pairwise different, since p is the longest simple path. Moreover, we have $q_{xyz} = q$. Since A is bideterministic by removing the non-accepting sink state, the state q_{xy} is also the uniquely reachable state from q by reading z backwards. Hence, the word xy^0z does not reach state q from the initial state q_0 because $q_x \neq q_{xy}$. Thus, the word w of length ℓ_A

cannot be pumped by respecting the equivalence classes of L because by assumption A is minimal and the states are representatives of the Myhill-Nerode equivalence classes. Hence, this shows that $\mathtt{mpe}(L) \geq \ell_A + 1 > \ell_A$.

2. The longest path p ends in the non-accepting sink state. Then the non-accepting sink state has a predecessor state in the longest path and the word w induced by the longest path shortened by the last step cannot be pumped which can be shown by a similar argument as above. Since $|w| = \ell_A - 1$, we find that $\mathtt{mpe}(L) \geq |w| + 1 = (\ell_A - 1) + 1 = \ell_A$.

In both cases we have shown that $\mathtt{mpe}(L) \geq \ell_A$. Together with the upper bound the stated claim $\ell_A \leq \mathtt{mpe}(L) \leq \ell_A + 1$ follows. □

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 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 mpe and 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.*

Proof. The proof of the statement follows similar lines as the proof of Lemma 19. The longest simple path between the initial state of the automaton to its unique accepting state gives rise to a word that cannot be pumped due to the uniqueness of the intermediate states due to the determinism and co-determinism of the accepting device (without the non-accepting sink state). Thus, one deduces $\mathtt{mpc}(L) \geq \ell_A\big|_{q_f} + 1$. On the other hand, any word that is accepted by A and is longer or equal than $\ell_A\big|_{q_f} + 1$ induces a computation where at least one state is repeated. Hence, the word can be pumped without losing its acceptance. Therefore, $\mathtt{mpc}(L) = \ell_A\big|_{q_f} + 1$ follows. □

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 P = NP. Also, assuming the Exponential Time Hypothesis (ETH), which is a stronger assumption than P ≠ 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[5] 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 P = NP. Assuming ETH, the inapproximability factor becomes $\omega(\frac{s \log\log s}{(\log s)^2})$.*

Proof. The following reduction shows that the problem s-t-LDP-problem, which is looking for the longest directed path that starts in a specified vertex s and ends in a specified vertex t, shares the same inapproximability properties: let G be an instance of the LDP-problem. We add two new vertices s and t and add directed edges from s to each vertex in G, as well as directed

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

edges from each vertex in G to the target t. Obviously the length of the longest directed s-t-path in the constructed graph thus obtained exceeds the length of the longest directed path in G exactly by two.

Next we reduce the s-t-LDP-problem to the Pumping-Problem of DFAs. Let $G = (V, E)$ be an instance of the s-t-LDP-problem with bounded outdegree. W.l.o.g. assume that the set V of vertices is equal to $V = \{v_1, v_2, \ldots, v_n\}$. We construct a partial DFA $A = (Q, \Sigma, \cdot, q_0, \{q_f\})$ with state set $Q = V$, input alphabet $\Sigma = \{\, a_{ij} \mid (v_i, v_j) \in E \,\}$, and transition function $v_i \cdot a_{ij} = v_j$, for each edge $(v_i, v_j) \in E$. The initial state is set to $q_0 = s$ and $q_f = t$ is the sole accepting state. Observe, that by construction the automaton is in fact even bideterministic. Also, we have $|\Sigma| = O(|Q|)$ because G has bounded outdegree.

Let $L := L(A)$. By Lemma 20 we obtain that $\mathtt{mpc}(L) = \ell_A\big|_{q_f} + 1$, which is one off the length of the longest directed s-t path in G. Thus, the desired inapproximability bounds follow by the results from [2]. A similar argumentation applies for the mpe-constant with the help of Lemma 19. □

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 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 the r-letter alphabet $\Sigma = \{a_1, a_2, \ldots, a_r\}$ and h be the homomorphism given by $a_i \mapsto \mathrm{bin}(i)\mathrm{bin}(i)^R$, for $1 \leq i \leq r$. Here, $\mathrm{bin}(i)$ denotes the $\lceil \log r \rceil$-bit binary encoding of the number i and $\mathrm{bin}(i)^R$ 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*

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

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

Proof. We will first show that $\mathtt{mpc}(h(L)) \leq b \cdot \mathtt{mpc}(L) - (b-1)$. In fact, a more general statement holds. Let Σ, Γ be two alphabets, and let g be a homomorphism $\Sigma^* \to \Gamma^*$. Then we have that $\mathtt{mpc}(g(L)) \leq \ell \cdot \mathtt{mpc}(L)$, where $\ell = \max_{a \in \Sigma} |g(a)|$. If w is a word in L which can be pumped with respect to L, and $w = xyz$ is a pumpable decomposition, then we also have $g(w) \in g(L)$ and for each integer $i \geq 0$ it holds that $g(x)g(y^i)g(z) = g(x)g(y)^i g(z) \in g(L)$. Every word in $g(L)$ of length at least $\ell \cdot \mathtt{mpc}(L)$ is the homomorphic image of a word of length at least $\mathtt{mpc}(L)$, and by the argument above is pumpable with respect to $g(L)$. For the homomorphism h under consideration, the above argument yields $\mathtt{mpc}(h(L)) \leq 2b \cdot \mathtt{mpc}(L)$. To see that in fact $\mathtt{mpc}(h(L)) \leq 2b \cdot \mathtt{mpc}(L) - (2b-1)$, we notice that all words in $h(L)$ are of some length which is a multiple of $2b$. Thus, *a forteriori* all words in $h(L)$, whose length is in the interval $[2b \cdot \mathtt{mpc}(L) - (2b-1), 2b \cdot \mathtt{mpc}(L) - 1]$ can be pumped, as there are no words at all that satisfy these conditions.

It remains to show that $\mathtt{mpc}(h(L)) \geq 2b \cdot \mathtt{mpc}(L) - 2b + 1$. It is readily verified that h is injective, prefix-free and suffix-free, and that it has the non-crossing property. Thus, h is a star-height preserving homomorphism. Now let u be a word in L that cannot be pumped with respect to L, and which furthermore satisfies $|u| = \mathtt{mpc}(L) - 1$. By Lemma 9, $h(u)$ cannot be pumped with respect to $h(L)$, which shows that $\mathtt{mpc}(h(L)) > 2b \cdot \mathtt{mpc}(L) - 2b$, as desired. □

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

Proof. As for the pumping constant $\mathtt{mpc}$ it holds for all alphabets Γ and all homomorphisms $g : \Sigma^* \to \Gamma^*$ that $\mathtt{mpe}(g(L)) \leq \ell \cdot \mathtt{mpe}(L)$, where $\ell = \max_{a \in \Sigma} |g(a)|$. Hence, we obtain the upper bound for $\mathtt{mpe}(h(L))$.

For the lower bound we argue as follows: by the definition of the minimal pumping constant $\mathtt{mpe}(L)$, for a language L, it immediatly follows that there is a word w (not necessarily contained in L) of length $\mathtt{mpe}(L) - 1$ that cannot be pumped w. r. t. Lemma 2. Then by the generalization of Lemma 9 to enhanced pumping we obtain that $h(w)$ cannot be pumped w. r. t. Lemma 2 either. Thus, we get $\mathtt{mpe}(h(L)) > 2b \cdot (\mathtt{mpe}(L) - 1)$. Hence that stated lower bound follows. □

One might ask whether we can come up with an exact calculation of $\mathtt{mpe}(h(L))$ as 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 the language $h(L) = (000001101001)^*$. 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 the inequality

$$\mathtt{mpe}(h(L)) = 13 > 12 + 1 = 4 \cdot 3 + 1 = 4 \cdot (4 - 1) + 1 = 2b \cdot (\mathtt{mpe}(L) - 1) + 1.$$

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.*

Proof. Let $G = (V, E)$ be the digraph from the reduction in the proof of Theorem 21. For the alphabet Σ of the DFA A constructed in Theorem 21, we use $a_1, a_2, \ldots, a_m$ to denote the symbols of Σ, with $m = |\Sigma|$. Recall that the code given by $h : a_i \mapsto \mathrm{bin}(i)(\mathrm{bin}(i))^R$, for $1 \leq i \leq m$, preserves the (enhanced) pumping property.

To construct a DFA accepting $h(L(A))$, we first construct an NFA as an intermediate step, and then show how we can get rid again of the nondeterministic choices. The NFA is constructed in polynomial time in a straightforward manner: we add directed paths from each vertex v, along which each codeword w can be read that encodes an edge e leaving v. Letting node v' denote the tip of edge $e = (v, v')$, the end of the directed path that is traversed by reading w is then identified with the state v' in the resulting NFA. This construction introduces $|w| - 1$ new states for this edge. Since each codeword w is of length $2 \log m$ and $m = O(n)$ because of the outdegree bound, the overall number of states s in the NFA thus constructed is at most $n + 2m \log m = O(n \log n)$. For the newly introduced paths leaving each vertex v, we can factor out common prefixes, and thus merge some of the states we just introduced. At the same time, the NFA becomes deterministic with this *manoeuvre*.

By virtue of Lemma 22, the minimal pumping constant w. r. t. Lemma 1 is increased by a factor $\sim 2 \log m$. The same holds true if we consider pumping w. r. t. Lemma 2. Thus, in both cases the calculation to come are the same. Thus, we only focus on the inapproximability of the minimal pumping constant w. r. t. Lemma 1.

For the sake of contradiction, assume now there is a polynomial time approximation algorithm for the pumping constant with a performance guarantee of $s^{1-\varepsilon}$, for some constant $\varepsilon > 0$. That is, the algorithm outputs solution size opt_1^*, which is at most $s^{1-\varepsilon}$ times larger than the optimum solution size opt_1. In terms of the parameter n, we thus obtain an approximation guarantee of

$$\frac{\mathrm{opt}_1^*}{\mathrm{opt}_1} \sim s^{1-\varepsilon} = O\left((n \log n)^{1-\varepsilon}\right).$$

If the pumping constant is at least opt_1^*, this bounds the size of the longest directed s-t path in the directed graph G by $\mathrm{opt}_2^* \sim \mathrm{opt}_1^* \cdot 2 \log m$. Since for the actual length opt_2 of a longest directed s-t path in G holds $\mathrm{opt}_2 \sim \mathrm{opt}_1 \cdot 2 \log m$. Thus, $\frac{\mathrm{opt}_2^*}{\mathrm{opt}_2} \sim \frac{\mathrm{opt}_1^*}{\mathrm{opt}_1}$, and the same approximation guarantee holds for the s-t-LDP problem. For every positive constant $\delta < \varepsilon$, we have $O\left((n \log n)^{1-\varepsilon}\right) = o(n^{1-\delta})$. Therefore, there is a polynomial time approximation algorithm approximating the length of the longest directed s-t path within $n^{1-\delta}$ for n large enough, which implies $\mathsf{P} = \mathsf{NP}$.

For the ETH-hardness bound, we argue along the same lines, but with a different calculation. Assume there is an approximation algorithm for the pumping constant running in polynomial time with a performance guarantee of $\frac{s \log \log s}{(\log s)^2}$. That is, the putative algorithm outputs solution size opt_1^* which is at most $\frac{s \log \log s}{(\log s)^2}$ times larger than the optimum solution size opt_1. We thus have an approximation guarantee of

$$\frac{\mathrm{opt}_1^*}{\mathrm{opt}_1} \sim \frac{s \log \log s}{(\log s)^2} = \frac{O(n \log n) \log \log s}{(\log s)^2} = O\left(\frac{n \log \log n}{\log n}\right).$$

In the last step, we used the fact that $\log s \sim \log n$ and $\log \log s \sim \log \log n$. In the same way as we did before, we can deduce a bound opt_2^* on the length of a longest directed s-t path in the digraph G from opt_1^*, with an approximation ratio satisfying $\frac{\mathrm{opt}_2^*}{\mathrm{opt}_2} \sim \frac{\mathrm{opt}_1^*}{\mathrm{opt}_1}$. We thus have obtained an approximation ratio $\frac{\mathrm{opt}_2^*}{\mathrm{opt}_2} = O\left(\frac{n \log \log n}{\log n}\right)$ for the s-t-LDP problem within polynomial time, which contradicts the Exponential Time Hypothesis by the result in [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.

Acknowledgements

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

References

1. A. Björklund. Personal communication. April 2024.
2. A. Björklund, T. Husfeldt, and S. Khanna. Approximating longest directed paths and cycles. In J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *Proceedings of the* 31*st International Colloquim Automata, Languages and Programming*, number 3142 in LNCS, pages 222–233, Turku, Finland, Juli 2004. Springer.
3. M. Cygan, F. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*, chapter Lower Bounds Based on the Exponential-Time Hypothesis, pages 467–521. Springer, 2015.
4. W. Czerwinski, M. Debski, T. Gogasz, G. Hoi, S. Jain, M. Skrzypczak, F. Stephan, and Ch. Tan. Languages given by finite automata over the unary alphabet. In P. Bouyer and S. Srinivasan, editors, *Proceedings of the* 43*rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023)*, volume 284 of *LIPIcs*, pages 22:1–22:20, IIIT Hyderabad, Telangana, India, Dezember 2023. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany.
5. J. Dassow and I. Jecker. Operational complexity and pumping lemmas. *Acta Inform.*, 59:337–355, 2022.
6. H. Fernau and A. Krebs. Problems on finite automata and the exponential time hypothesis. *Algorithms*, 10(1):24, 2017.
7. H. Gruber and M. Holzer. Tight bounds on the descriptional complexity of regular expressions. In V. Diekert and D. Nowotka, editors, *Proceedings of the* 13*th International Conference Developments in Language Theory*, number 5583 in LNCS, pages 276–287, Stuttgart, Germany, Juni–Juli 2009. Springer.
8. H. Gruber, M. Holzer, and C. Rauch. The pumping lemma for regular languages is hard. In B. Nagy, editor, *Proceedings of the* 27*th International Conference on Implementation and Application of Automata*, number 14151 in LNCS, pages 128–140, Famagusta, Cyprus, September 2023. Springer.
9. H. Gruber, M. Holzer, and S. Wolfsteiner. On minimizing regular expressions without kleene star. In E. Bampis and A. Pagourtzis, editors, *Proceedings of the* 23*rd International Symposium on Fundamentals of Computation Theory*, number 12867 in LNCS, pages 245–258, Athens, Greece, September 2021. Springer.
10. M. A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, 1978.
11. K. Hashiguchi and N. Honda. Homomorphisms that preserve star height. *Inform. Comput.*, 30(3):247–266, 1976.
12. M. Holzer and C. Rauch. On Jaffe's pumping lemma, revisited. In H. Bordihn, N. Tran, and G. Vaszil, editors, *Proceedings of the* 25*th International Conference on Descriptional Complexity of Formal Systems*, number 13918 in LNCS, pages 65–78, Potsdam, Germany, Juli 2023. Springer.
13. R. Impagliazzo and R. Paturi. Complexity of k-SAT. In B. Sirakov, P. N. de Souza, and M. Viana, editors, *Proceedings of the* 14*th Annual IEEE Conference on Computational Complexity*, pages 237–240, Atlanta, Georgia, USA, Mai 1999. IEEE Computer Society.
14. J. Jaffe. A necessary and sufficient pumping lemma for regular languages. *SIGACT News*, 10(2):48–49, Sommer 1978.
15. D. C. Kozen. *Automata and Computability*. Undergraduate Texts in Computer Science. Springer, 1997.
16. D. Lakshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the European Association for Theoretical Computer Science*, 105:41–72, Oktober 2011.
17. R. McNaughton. The loop complexity of regular events. *Information Sciences*, 1:305–328, 1969.
18. A. Nijholt. YABBER—yet another bibliography: Pumping lemma's. An annotated bibliography of pumping. *Bull. EATCS*, 17:34–53, 1982.

19. J.-E. Pin. On reversible automata. In *Proceedings of* 1*st Latin American Symposium on Theoretical Informatics*, number 583 in LNCS, pages 401–416, São Paulo, Brazil, April 1992. Springer.
20. M. P. Schützenberger. Une théorie algébrique du codage. *Séminaire Dubreil. Algèbre et théorie des nombres*, 9:1–24, 1955-1956.
21. V. Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In B. Sirakov, P. N. de Souza, and M. Viana, editors, *Proceedings of the International Congress of Mathematicians*, pages 3447–3487, Rio de Janeiro, Brazil, April 2019. World Scientific.

Institut für Informatik

Justus-Liebig-Universität Giessen
Arndtstr. 2, 35392 Giessen, Germany

Recent Reports

(Further reports are available at `www.informatik.uni-giessen.de`.)

M. Holzer, *Comments on Monoids Induced by NFAs*, Report 2301, March 2023.

S. Beier, M. Holzer, *Semi-Linear Lattices and Right One-Way Jumping Finite Automata*, Report 1901, April 2019.

M. Kutrib, T. Worsch, *Self-Verifying Cellular Automata*, Report 1803, April 2018.

S. Beier, M. Holzer, *Properties of Right One-Way Jumping Finite Automata*, Report 1802, March 2018.

B. Truthe, *Hierarchy of Subregular Language Families*, Report 1801, February 2018.

M. Holzer, M. Hospodár, *On the Magic Number Problem of the Cut Operation*, Report 1703, October 2017.

M. Holzer, S. Jakobi, *A Note on the Computational Complexity of Some Problems for Self-Verifying Finite Automata*, Report 1702, April 2017.

S. Beier, M. Holzer, M. Kutrib, *On the Descriptional Complexity of Operations on Semilinear Sets*, Report 1701, April 2017.

M. Holzer, S. Jakobi, M. Wendlandt, *On the Computational Complexity of Partial Word Automata Problems*, Report 1404, May 2014.

H. Gruber, M. Holzer, *Regular Expressions From Deterministic Finite Automata, Revisited*, Report 1403, May 2014.

M. Kutrib, A. Malcher, M. Wendlandt, *Deterministic Set Automata*, Report 1402, April 2014.

M. Holzer, S. Jakobi, *Minimal and Hyper-Minimal Biautomata*, Report 1401, March 2014.

J. Kari, M. Kutrib, A. Malcher (Eds.), *19th International Workshop on Cellular Automata and Discrete Complex Systems AUTOMATA 2013 Exploratory Papers*, Report 1302, September 2013.

M. Holzer, S. Jakobi, *Minimization, Characterizations, and Nondeterminism for Biautomata*, Report 1301, April 2013.

A. Malcher, K. Meckel, C. Mereghetti, B. Palano, *Descriptional Complexity of Pushdown Store Languages*, Report 1203, May 2012.

M. Holzer, S. Jakobi, *On the Complexity of Rolling Block and Alice Mazes*, Report 1202, March 2012.

M. Holzer, S. Jakobi, *Grid Graphs with Diagonal Edges and the Complexity of Xmas Mazes*, Report 1201, January 2012.

H. Gruber, S. Gulan, *Simplifying Regular Expressions: A Quantitative Perspective*, Report 0904, August 2009.

M. Kutrib, A. Malcher, *Cellular Automata with Sparse Communication*, Report 0903, May 2009.

M. Holzer, A. Maletti, *An $n \log n$ Algorithm for Hyper-Minimizing States in a (Minimized) Deterministic Automaton*, Report 0902, April 2009.

H. Gruber, M. Holzer, *Tight Bounds on the Descriptional Complexity of Regular Expressions*, Report 0901, February 2009.

M. Holzer, M. Kutrib, and A. Malcher (Eds.), *18. Theorietag Automaten und Formale Sprachen*, Report 0801, September 2008.

M. Holzer, M. Kutrib, *Flip-Pushdown Automata: Nondeterminism is Better than Determinism*, Report 0301, February 2003

M. Holzer, M. Kutrib, *Flip-Pushdown Automata: $k+1$ Pushdown Reversals are Better Than k*, Report 0206, November 2002

M. Holzer, M. Kutrib, *Nondeterministic Descriptional Complexity of Regular Languages*, Report 0205, September 2002

H. Bordihn, M. Holzer, M. Kutrib, *Economy of Description for Basic Constructions on Rational Transductions*, Report 0204, July 2002

M. Kutrib, J.-T. Löwe, *String Transformation for n-dimensional Image Compression*, Report 0203, May 2002

A. Klein, M. Kutrib, *Grammars with Scattered Nonterminals*, Report 0202, February 2002

A. Klein, M. Kutrib, *Self-Assembling Finite Automata*, Report 0201, January 2002

M. Holzer, M. Kutrib, *Unary Language Operations and its Nondeterministic State Complexity*, Report 0107, November 2001

A. Klein, M. Kutrib, *Fast One-Way Cellular Automata*, Report 0106, September 2001