On Minimal Grammar Problems for Finite Languages (Extended Abstract)

Hermann Gruber[1], Markus Holzer[2], and Simon Wolfsteiner[3]*

[1] knowledgepark GmbH,
Leonrodstr. 68, 80636 München, Germany
`hermann.gruber@kpark.de`

[2] Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
`holzer@informatik.uni-giessen.de`

[3] Institut für Diskrete Mathematik und Geometrie, TU Wien,
Wiedner Hauptstr. 8–10, 1040 Wien, Austria
`simon.wolfsteiner@tuwien.ac.at`

Abstract. We investigate the grammatical complexity of finite languages w.r.t. context-free grammars and variants thereof. For fixed alphabets, it is shown that both the minimal number of productions and the minimal size of a context-free grammar generating a finite language cannot be approximated within a factor of $o(p^{1/6})$ and $o(s^{1/7})$, respectively, unless $\mathsf{P} = \mathsf{NP}$. Here, p is the number of productions and s the size of the given grammar. Similar inapproximability results also hold for linear context-free and right-linear (or regular) grammars. As a byproduct, we show that the language of all cubes of a given length requires an exponential number of context-free productions and we also investigate upper and lower bounds on the complexity of the operations union and concatenation for finite languages.

1 Introduction

Questions regarding the economy of descriptions of formal languages by different formalisms such as automata, grammars, and formal systems have been studied quite extensively in the past, see, e.g., [13]. The results in [2, 4] mark the starting point of a theory of the grammatical complexity of finite languages, where the chosen complexity measure is the number of productions. In particular, [4] gives a relative succinctness classification for various kinds of context-free grammars. Further results along these lines can be found in [1–3, 16] as well as some newer ones in, e.g., [7, 8, 10]. It is worth mentioning that in [10] a method for proving lower bounds on the number of productions for context-free grammars was developed. For instance, it was shown that any context-free grammar generating the set of all squares of a given length requires an exponential number of productions.

* This research was completed while the author was on leave at the Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany, in 2017 and is supported by the Vienna Science Fund (WWTF) project VRG12-004.

More recently, it was shown that there is a close relationship between a certain class of formal proofs in first-order logic and a certain class of (tree) grammars. In particular, the number of productions in such a grammar corresponds to the number of certain inference rules in the proof [9]. This correspondence sparked our interest in further investigating questions regarding the grammatical complexity of finite languages. The main result of this paper is that, for fixed alphabets of size at least 5, the minimal number of productions necessary to generate a finite language by a context-free grammar cannot be approximated within a factor of $o(p^{1/6})$, unless $\mathsf{P} = \mathsf{NP}$. Here, p is the number of productions in the given grammar. Since the size of a context-free grammar depends on the number of its productions, the above result also implies that the size of a minimal grammar generating a finite language cannot be approximated within a factor of $o(s^{1/7})$, unless $\mathsf{P} = \mathsf{NP}$, where s is the size of the given grammar. This second result is related to the NP-hard smallest grammar problem with approximation ratio at least $\frac{8569}{8568}$, unless $\mathsf{P} = \mathsf{NP}$ from [6]. There, the smallest grammar problem asks for the smallest (in terms of size) context-free grammar that generates exactly *one* given word. As a byproduct of our inapproximability result, we show—using elementary methods already developed in [4]—that the set of all cubes of a given length requires an exponential number of productions. To be more precise, the language $T_n = \{ w\$w\#w \mid w \in \{0,1\}^n \}$ requires exactly 2^n context-free productions; this is also a lower bound on the size of a grammar for T_n. This result is more precise than using the lower bound method from [10] that results only in a lower bound of $\Omega(2^{n/8}/\sqrt{3n})$ many context-free productions. To obtain our main result, we reduce from the coNP-complete propositional *tautology* problem. The reduction features a gadget based on the language T_n. The correctness proof of the reduction uses estimates on the grammatical complexity of language operations on finite languages. Therefore, we study the grammatical complexity of union and concatenation of finite languages, thus complementing the results from [7, 8] on infinite languages. While for union we have a tight upper bound for all grammar types under consideration, the situation changes when we consider concatenation. Here, the upper bound for linear grammars differs from the one for the other grammar types. However, so far, we were not able to show that the upper bound is tight for any of the considered grammar types.

2 Preliminaries

We assume that the reader is familiar with the basic notions of formal language theory as contained in [12]. Nevertheless, to fix notation and terminology, we introduce the basic notions and results relevant to this paper in this section.

Let Σ be a finite alphabet. Then Σ^* denotes the set of all words over the finite alphabet Σ including the *empty word* ε, and we write Σ^+ for $\Sigma^* \setminus \{\varepsilon\}$. The *length* of a word w in Σ^* is denoted by $|w|$. In particular, the length of the empty word ε is 0, i.e., $|\varepsilon| = 0$. Let $\ell \geq 0$. Then Σ^ℓ ($\Sigma^{\leq \ell}$, respectively) refers to the set of all words over Σ of length exactly ℓ (at most ℓ, respectively). A subset L of Σ^* is said to be a *language*. Any language $L \subseteq \Sigma^{\leq \ell}$, for $\ell \geq 0$, is called *finite* and, unless stated otherwise, we always assume that $\ell = \max\{\, |w| \mid w \in L \,\}$. If L is a subset of Σ^ℓ, for $\ell \geq 0$, then L is called a *uniform language*. This means that in a uniform language all words have the same length.

A *context-free grammar* (CFG) is a 4-tuple $G = (N, \Sigma, P, S)$, where N and Σ are disjoint alphabets of *nonterminals* and *terminals*, respectively, $S \in N$ is the *axiom* (or *start symbol*), and P is a finite set of *productions* of the form $A \to \alpha$, where $A \in N$ and $\alpha \in (N \cup \Sigma)^*$. As usual, the derivation relation of G is denoted by $\Rightarrow_G$ and the reflexive and transitive closure of $\Rightarrow_G$ is written as $\Rightarrow_G^*$. The *language generated* by G is defined as $L(G) = \{\, w \in \Sigma^* \mid S \Rightarrow_G^* w \,\}$. We also consider the following restrictions of context-free (CF) grammars: (i) a context-free grammar is said to be *linear context-free* (LIN) if the productions are of the form $A \to \alpha$, where $A \in N$ and $\alpha \in \Sigma^*(N \cup \{\varepsilon\})\Sigma^*$, and (ii) a context-free grammar is said to be *right-linear* or *regular* (REG) if the productions are of the form $A \to \alpha$, where $A \in N$ and $\alpha \in \Sigma^*(N \cup \{\varepsilon\})$. Furthermore, Γ will denote the set of those abbreviations in the sequel, that is, $\Gamma = \{\mathsf{REG}, \mathsf{LIN}, \mathsf{CF}\}$.

We are interested in the complexity of finite languages w.r.t. different grammar types. To be more precise: what is the smallest number of productions a grammar needs in order to generate a language L? Let $G = (N, \Sigma, P, S)$ be a context-free grammar. We define $|G|$ to be the number of productions if not stated otherwise, i.e., $|G| = |P|$. Then the *complexity* of a finite language L w.r.t. X-grammars, for $X \in \Gamma$, also called the X-complexity of L, is defined as

$$\mathsf{Xc}(L) = \min\{\, |G| \mid G \text{ is an } X\text{-grammar with } L = L(G) \,\}.$$

We say that G is a *minimal* X-grammar, for $X \in \Gamma$, generating a finite language L if $L(G) = L$ and $|G| = \mathsf{Xc}(L)$.

The previously introduced measures for the different types of grammars are related to each other as follows: by definition, $\mathsf{CF} \leq_c \mathsf{LIN} \leq_c \mathsf{REG}$, where $X \leq_c Y$, for $X, Y \in \Gamma$, if and only if $\mathsf{Xc}(L) \leq \mathsf{Yc}(L)$, for every finite language L. In the case that $X \leq_c Y$, we say that X is *more succinct than* Y.

3 Incompressible Languages

In this section, we consider different finite languages and show that they can only be generated minimally by listing all words that belong to the language under consideration. Such languages will be called *(context-free) incompressible languages* in the following. Some incompressible languages can already be found in the seminal papers [2, 4] on concise description of finite languages by different types of grammars. For instance, as shown in [2], the language

$$L_n = \{\, a^i b^i c^i \mid 1 \leq i \leq n \,\}$$

contains a linear number of words and satisfies $\mathsf{CFc}(L_n) = |L_n|$. Further examples of incompressible languages can be found in [4]. The proof is based on the following Lemma 1 which states some easy facts about *minimal* context-free grammars generating finite languages and was established in [4].

Lemma 1. *Let $G = (N, \Sigma, P, S)$ be a minimal context-free grammar generating a finite language L. Then (i) for every $A \in N \setminus \{S\}$, there are $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$ with $\alpha_1 \neq \alpha_2$ such that $A \to \alpha_1$ and $A \to \alpha_2$ are in P, (ii) for every $A \in N \setminus \{S\}$, the set $L_A(G) = \{\, w \in \Sigma^* \mid A \Rightarrow_G^* w \,\}$ contains at least two words, (iii) there is no derivation of the form $A \Rightarrow_G^+ \alpha_1 A \alpha_2$ with $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$, and (iv) for every nonterminal $A \in N \setminus \{S\}$, there are $u_1, u_2, v_1, v_2 \in \Sigma^*$ such that $u_1 A u_2 \neq v_1 A v_2$ and $S \Rightarrow_G^* u_1 A u_2$ and $S \Rightarrow_G v_1 A v_2$.*

A closer look at [4] reveals that the incompressible languages presented there are of size polynomial in the length of a longest word. But what about incompressible finite languages containing many short words? In fact, for some languages it is known that the CF-complexity is high. For instance, recently in [10], it was shown that any context-free grammar for the copy language $C_n = \{\, w\$w \mid w \in \{0,1\}^n \,\}$ over a two letter alphabet has size at least $\Omega(2^{n/4}/\sqrt{2n})$—there, the size was alternatively defined as the sum of all production lengths; the length of a production $A \to \alpha$ is $|\alpha| + 2$. The technique presented in [10] to prove this result is quite involved and generalises a previously known result on an exponential lower bound on context-free grammars generating the set of all permutations over a finite alphabet. Note that the above mentioned lower bound for the language C_n is *not* enough to prove that this language is incompressible in our sense. In [10], also more complicated languages such as

$$T_n = \{\, w\$w\#w \mid w \in \{0,1\}^n \,\},$$

the language of all triples of length n, are considered. The lower bound for T_n obtained in [10] is $\Omega(2^{n/8}/\sqrt{3n})$, while the trivial upper bound is $|T_n| = 2^n$. In contrast, we are able to show that T_n is generated minimally by a context-free grammar only by simply listing all words. Notably, this can be derived using the classic technique from [4].

Theorem 2. *Let $X \in \Gamma$ and $n \geq 1$. Then $\mathsf{Xc}(T_n) = |T_n|$.*

Proof. theorem Let $G = (N, \Sigma, P, S)$ be a minimal CFG generating T_n. Moreover, let the nonterminal $A \in N \setminus \{S\}$ be arbitrary. By Lemma 1, there are derivations $S \Rightarrow_G^* u_1 A u_2$ and $S \Rightarrow_G^* w_1 A w_2$ with $u_1, u_2, w_1, w_2 \in \Sigma^*$ and $u_1 A u_2 \neq w_1 A w_2$ as well as $x, y \in \Sigma^*$ with $x \neq y$, $A \Rightarrow_G^* x$, and $A \Rightarrow_G^* y$. We will first show that it is impossible that $x, y \in \{0,1\}^*$ and then that both x and y must contain the symbols \$ and #. Note that $|x| = |y|$, for otherwise one could derive words v_1 and v_2 such that $|v_1| \neq |v_2|$, but T_n only contains words having the same length.

Assume, w.l.o.g., that $x \in \{0,1\}^*$ (the case that $y \in \{0,1\}^*$ is symmetric). From $x \neq y$ and $|x| = |y|$ it follows that both $x \neq \varepsilon$ and $y \neq \varepsilon$ must hold. Now, let $w \in \{0,1\}^n$; we distinguish three cases:

1. Suppose that $u_1 x \in \{0,1\}^*$ and $u_2 \in \{0,1\}^* \{\$\} \{w\} \{\#\} \{w\}$. Then $u_1 x u_2 = v_1 \$w\#w$ and $u_1 y u_2 = v_2 \$w\#w$, for $v_1, v_2 \in \{0,1\}^n$ and $v_1 \neq v_2$. Thus, either $v_1 \neq w$ or $v_2 \neq w$, i.e., $v_1\$w\#w \notin T_n$ or $v_2\$w\#w \notin T_n$. This is a contradiction.

2. Suppose that $u_1 \in \{w\}\{\$\}\{0,1\}^*$ and $u_2 \in \{0,1\}^*\{\#\}\{w\}$. Then, since $x \neq y$, there are two derivations $S \Rightarrow_G^* u_1Au_2 \Rightarrow_G^* v = u_1xu_2$ and $S \Rightarrow_G^* u_1Au_2 \Rightarrow_G^* v' = u_1yu_2$ with $v \neq v'$. In particular, $v = w\$u\#w$ and $v' = w\$u'\#w$ with $u \neq w$ or $u' \neq w$. But this means that $v \notin T_n$ or $v' \notin T_n$, which is a contradiction.
3. Suppose that $u_1 \in \{w\}\{\$\}\{w\}\{\#\}\{0,1\}^*$ and $xu_2 \in \{0,1\}^*$. Symmetric to case 1. Thus, we obtain a contradiction again.

Similar arguments show that either $w_1xw_2 \notin T_n$ or $w_1yw_2 \notin T_n$. Hence, we have both $x \notin \{0,1\}^*$ and $y \notin \{0,1\}^*$. Now, suppose that x or y does not contain both \$ and #. Assume, w.l.o.g., that x contains # but does not contain \$. Let $w \in \{0,1\}^n$; we distinguish two cases:

1. Suppose that y contains \$. Then we have $S \Rightarrow_G^* u_1Au_2 \Rightarrow_G^* v = u_1xu_2$ and $S \Rightarrow_G^* u_1Au_2 \Rightarrow_G^* v' = u_1yu_2$, where either v does not contain \$ or v' contains at least two occurrences of \$. Thus, either $v \notin T_n$ or $v' \notin T_n$.
2. Suppose that y contains # . Then we have $S \Rightarrow_G^* u_1Au_2 \Rightarrow_G^* v = u_1xu_2$ and $S \Rightarrow_G^* u_1Au_2 \Rightarrow_G^* v' = u_1yu_2$. Together with the fact that $x \neq y$ and $|x| = |y|$, it follows that if $v \in T_n$, then $v' \notin T_n$. If, on the other hand, $v' \in T_n$, we have $v \notin T_n$.

In both of these cases, we obtain a contradiction. Thus, both words x and y must contain both \$ and #. This and $|x| = |y|$ implies that $x = y$. This is a contradiction to our assumption. Hence, $N = \{S\}$ and therefore the only way to generate the language T_n is to list all of its words as right-hand sides of productions having the sole nonterminal S on the left-hand side. Since by definition $\mathsf{CF} \leq_c \mathsf{LIN} \leq_c \mathsf{REG}$, the statement also holds for $X \in \{\mathsf{REG}, \mathsf{LIN}\}$. □

Later, we will use the language T_n as one of our basic building blocks for the inapproximability result of the exact complexity of finite languages. It is worth mentioning that the question as to whether the copy language C_n is incompressible, i.e., whether it can only be generated minimally by a context-free grammar by listing all words, is still open.

4 Language Operations

Now, we turn our attention to the exact complexity of two operations on finite languages, namely *union* and *concatenation*. The reason for specifically considering these two operations is due to the fact that the language which we use in our reduction is defined in terms of union and concatenation and, for our inapproximability result, we need to obtain complexity bounds on context-free grammars. In [7, 8], the authors considered some related problems on union and concatenation, where they discussed the range of applying the respective operation to two (or a finite number of) languages. They did not restrict themselves to finite languages, but they showed, e.g., that for two ε-free finite languages L_1 and L_2 defined over disjoint alphabets the following statement holds:

$$\mathsf{Xc}(L_1 \cup L_2) = \mathsf{Xc}(L_1) + \mathsf{Xc}(L_2) \quad \text{if } X \in \Gamma.$$

However, we will not restrict ourselves to the union of two finite languages with disjoint alphabets. Instead, we will show that the following upper bound holds for arbitrary finite languages L_1 and L_2 and that it is tight in the sense that we can give an example of a language that actually reaches this upper bound.

Theorem 3. *Let $X \in \Gamma$ and L_1 and L_2 be finite languages. Then it holds that $\mathsf{Xc}(L_1 \cup L_2) \leq \mathsf{Xc}(L_1) + \mathsf{Xc}(L_2)$.*

In order to show that the upper bound obtained in Theorem 3 is tight, we need to show the following Lemma 4—which states that if we decompose a finite incompressible language into a disjoint union of two languages, then these two disjoint languages must be incompressible as well.

Lemma 4. *Let $X \in \Gamma$ and L be a finite language with $\mathsf{Xc}(L) = |L|$. Moreover, let L_1 and L_2 be disjoint finite languages such that $L = L_1 \cup L_2$. Then we have $\mathsf{Xc}(L_1) = |L_1|$ and $\mathsf{Xc}(L_2) = |L_2|$.*

Recall the finite CF-incompressible language $L_n = \{\, a^i b^i c^i \mid 1 \leq i \leq n \,\}$ with $|L_n| = n$. Since L_n can be defined in terms of the union of two disjoint finite languages, we can use it in conjunction with Lemma 4 in order to show that the upper bound of Theorem 3 is tight:

Theorem 5. *Let $X \in \Gamma$. Then there is an alphabet Σ such that for all $m, n \geq 1$, there exist finite languages L_1 and L_2 over Σ with $\mathsf{Xc}(L_1) = m$ and $\mathsf{Xc}(L_2) = n$ such that $\mathsf{Xc}(L_1 \cup L_2) \geq \mathsf{Xc}(L_1) + \mathsf{Xc}(L_2)$.*

Proof. theorem Let $X \in \Gamma$ and $\Sigma = \{a, b, c\}$. For $m, n \geq 1$, define the finite language

$$L = \{\, a^i b^i c^i \mid 1 \leq i \leq m + n \,\}.$$

Moreover, let $L_1 = \{\, a^i b^i c^i \mid 1 \leq i \leq m \,\}$ and $L_2 = L \setminus L_1$. Clearly, the intersection satisfies $L_1 \cap L_2 = \emptyset$ and $L_1 \cup L_2 = L$. Since both languages L and L_1 are CF-incompressible, we have $\mathsf{CFc}(L) = |L| = m + n$ and $\mathsf{CFc}(L_1) = m$. Thus, it follows that $\mathsf{CFc}(L_2) = |L_2| = |L| - |L_1| = n$, by Lemma 4. Consequently,

$$\mathsf{CFc}(L_1 \cup L_2) = \mathsf{CFc}(L) = m + n = \mathsf{CFc}(L_1) + \mathsf{CFc}(L_2).$$

This finishes the proof of the stated claim. □

Let us now consider the concatenation of two finite languages. As we will see in the following, there are some subtleties that make things more complicated than in the case of union. A first difference is that there is no uniform upper bound for all grammar types in Γ. The main reason is that we have not yet found a better method for combining two linear grammars in a way that the resulting grammar is linear as well apart from pre- or appending a regular grammar generating one of the two languages (in a suitable fashion) to the given linear grammar generating the other one. For the other grammar types, we obtain an upper bound that corresponds to the sum of the respective complexities.

Theorem 6. *Let $X \in \{\mathsf{REG}, \mathsf{CF}\}$ and L_1 and L_2 be finite languages. Then*

1. $\mathsf{Xc}(L_1L_2) \leq \mathsf{Xc}(L_1) + \mathsf{Xc}(L_2)$,
2. $\mathsf{LINc}(L_1L_2) \leq \min\{\, \mathsf{REGc}(L_1) + \mathsf{LINc}(L_2), \mathsf{LINc}(L_1) + \mathsf{REGc}(L_2)\,\}$.

Another difference between union and concatenation is that we have not yet been able to show that the upper bound is tight, i.e., we do not know whether there exists a finite language that reaches the upper bound. The good news is, however, that we can prove a lower bound on the X-complexity, for $X \in \Gamma$, for the concatenation of finite languages. To obtain the lower bound, we will show—given two finite languages L_1 and L_2 as well as a grammar G that generates $L_1\#L_2$—how to construct a grammar G' from G that generates the quotient of $L_1\#$ with the language $L(G)$, that is, $L(G') = (L_1\#)^{-1}L(G) = L_2$. The left quotient of a language L_1 with a language L_2 is defined as follows:

$$L_1^{-1}L_2 = \{\, v \in \Sigma^* \mid \text{there is some } w \in L_1 \text{ s.t. } wv \in L_2\,\}.$$

We omit the braces if L_1 is a singleton, i.e., we write $w^{-1}L_2$ instead of $\{w\}^{-1}L_2$.

Lemma 7. *Let $X \in \Gamma$ and L_1 and L_2 be finite languages over the alphabet Σ. Then $\mathsf{Xc}(L_1\#L_2) \geq \max\{\, \mathsf{Xc}(L_1), \mathsf{Xc}(L_2)\,\}$, where $\#$ does not occur in Σ.*

We will now take a closer look at how the X-complexity of a language L changes when we append a fresh symbol to all words in L.

Lemma 8. *Let $X \in \Gamma$. Assume that L is a finite language over the alphabet Σ. Then $\mathsf{Xc}(L \cdot \#) = \mathsf{Xc}(L)$, where $\#$ is a letter that is not contained in Σ. The statement remains valid if one considers the language $\# \cdot L$ instead of $L \cdot \#$.*

As already mentioned, thus far we have not been able to show that the upper bound on the X-complexity of concatenation is tight, but an immediate consequence of Lemmata 7 and 8 are the following lower bounds on the concatenation of two finite languages:

Theorem 9. *Let $X \in \Gamma$. Then there is an alphabet Σ such that for all $m, n \geq 1$, there exist finite languages L_1 and L_2 over Σ with $\mathsf{Xc}(L_1) = m$ and $\mathsf{Xc}(L_2) = n$ such that $\mathsf{Xc}(L_1 \cdot L_2) \geq \max\{\, \mathsf{Xc}(L_1), \mathsf{Xc}(L_2)\,\}$.*

5 Inapproximability Results

In this section, we will show the main result of this paper, namely that the minimal number of context-free productions necessary to generate a finite language cannot be approximated within a certain factor, unless $\mathsf{P} = \mathsf{NP}$. The language T_n—introduced in Section 3—will be a basic building block for this endeavour. The proof strategy is by a reduction from the coNP-complete *tautology* problem for 3-DNF-formulae: given a formula φ with m conjunctive clauses and n variables, where each clause is the conjunction of at most 3 literals, it is coNP-complete to determine whether φ is a tautology—in other words, whether the negation $\neg\varphi$ of φ is *unsatisfiable*. Then the core idea is to give a suitable representation of the satisfying assignments of φ in $\{0,1\}^n$ for the n variables in form of a grammar G_φ such that φ is a tautology if and only if $L(G_\varphi) = \{0,1\}^n$; by construction, there is a one-to-one correspondence between assignments and words from the set $\{0,1\}^n$. In order to finish our reduction, we embed G_φ into a grammar that generates the language

$$L_\varphi = L(G_\varphi) \cdot \{\&\} \cdot \{0,1,\$,\#\}^{3c\cdot\lceil \log n\rceil+2} \cup \{0,1\}^n \cdot \{\&\} \cdot T_{c\cdot\lceil \log n\rceil},$$

for some carefully chosen constant c. It is not hard to see that this reduction is polynomial even if we force the grammar for L_φ to be regular. Then, we distinguish two cases: (i) On the one hand, if φ is a tautology, then we have $L_\varphi = \{0,1\}^n \cdot \{\&\} \cdot \{0,1,\$,\#\}^{3c\cdot\lceil \log n\rceil+2}$ and there is a context-free grammar with a constant number of productions that generates L_φ. For the X-grammars with $X \in \{\mathsf{REG}, \mathsf{LIN}\}$, a linear number of productions suffices, i.e., the number is in $O(n)$. (ii) On the other hand, if φ is not a tautology, then there is an assignment under which φ evaluates to *false*. Hence, there is a word $w \in \{0,1\}^n$ that corresponds to that assignment and is *not* a member of $L(G_\varphi)$. But then the left quotient of L_φ w.r.t. the word $w\&$, that is, the language $\{\, v \in \{0,1,\$,\#\}^* \mid w\&v \in L_\varphi \,\}$, is equal to the language of triples $T_{c\cdot\lceil \log n\rceil}$. From this quotient construction, it will then follow that $\mathsf{Xc}(T_{c\cdot\lceil \log n\rceil}) = O(\mathsf{Xc}(L_\varphi)\cdot n^4)$. Since we have already seen that $\mathsf{Xc}(T_{c\cdot\lceil \log n\rceil}) = \Omega(n^c)$, we can deduce that $\mathsf{Xc}(L_\varphi) = \Omega(n^{c-4})$. This allows us to prove the main result of this paper:

Theorem 10. *Let $X \in \Gamma$. Given an X-grammar with p productions generating a finite language L, it is impossible to approximate $\mathsf{Xc}(L)$ within a factor of $o(p^{1/6})$, unless $\mathsf{P} = \mathsf{NP}$.*

We assume that the reader is familiar with the syntax and semantics of propositional logic. In the following, we denote the set of *propositional variables* by $V = \{x_1, x_2, \ldots\}$. We write $\mathsf{var}(\varphi)$ for the set of variables occurring in a propositional formula φ and expressions of the form x and $\neg x$, for $x \in V$, are called *literals*. As usual, a *truth assignment* is a mapping $\sigma\colon V \to \{0,1\}$ which can be inductively extended to formulae. Since we are going to reduce propositional formulae to context-free grammars, we will also identify truth assignments for a formula φ with words in $\{0,1\}^n$: if $\mathsf{var}(\varphi) = \{x_1, x_2, \ldots x_n\}$, then the word corresponding to the truth assignment $\sigma(\varphi)$ is given by $\sigma(x_1, x_2, \ldots, x_n) := \sigma(x_1)\sigma(x_2)\ldots\sigma(x_n)$. Moreover, we write $\sigma \models \varphi$ and $\sigma(x_1)\sigma(x_2)\ldots\sigma(x_n) \models \varphi$ if $\mathsf{var}(\varphi) = \{x_1, x_2, \ldots, x_n\}$ and $\sigma(\varphi) = 1$.

For the sake of convenience, we will identify a clause $C = \ell_1 \wedge \ell_2 \wedge \ell_3$ also with a set of literals $C = \{\ell_1, \ell_2, \ell_3\}$.

Now, we will construct a regular grammar that generates the satisfying truth assignments of the given propositional formula φ in 3-DNF with $|\mathsf{var}(\varphi)| = n$, thus reducing the propositional tautology problem to the fixed length universality problem: does it hold that $L(G) = \Sigma^\ell$, for a given grammar G and an integer ℓ. Let φ be a propositional formula in 3-DNF with $\mathsf{var}(\varphi) = \{x_1, x_2, \ldots, x_n\}$ consisting of the conjunctive clauses $C_1, C_2, \ldots, C_m$, for $m \geq 1$. We define a right-linear grammar $G_\varphi = (N, \{0,1\}, P, S)$ as follows: the nonterminals are given by $N = \{S\} \cup \{ A_{i,j} \mid 1 \leq i \leq m, 2 \leq j \leq n \}$, and the set P consists of the following productions

- $A_{i,j} \to 1A_{i,j+1}$ whenever $x_j \in C_i$ and $\neg x_j \notin C_i$, for $1 \leq i \leq m, 1 \leq j \leq n-1$,
- $A_{i,j} \to 0A_{i,j+1}$ whenever $x_j \notin C_i$ and $\neg x_j \in C_i$, for $1 \leq i \leq m, 1 \leq j \leq n-1$,
- $A_{i,j} \to 1A_{i,j+1}$ whenever $x_j \notin C_i$ and $\neg x_j \notin C_i$, for $1 \leq i \leq m, 1 \leq j \leq n-1$,
- $A_{i,j} \to 0A_{i,j+1}$ whenever $x_j \notin C_i$ and $\neg x_j \notin C_i$, for $1 \leq i \leq m, 1 \leq j \leq n-1$,
- $A_{i,n} \to 1$ whenever $x_n \in C_i$ and $\neg x_n \notin C_i$, for $1 \leq i \leq m$,
- $A_{i,n} \to 0$ whenever $x_n \notin C_i$ and $\neg x_n \in C_i$, for $1 \leq i \leq m$,
- $A_{i,n} \to 1$ whenever $x_n \notin C_i$ and $\neg x_n \notin C_i$, for $1 \leq i \leq m$,
- $A_{i,n} \to 0$ whenever $x_n \notin C_i$ and $\neg x_n \notin C_i$, for $1 \leq i \leq m$,

where, for all $i \in \{1, 2, \ldots, m\}$, we set $A_{i,1} := S$. Note that the above reduction from 3-DNF formulae to context-free grammars is essentially the same as the reduction to regular expressions presented in [14].

The following proposition expresses that the number of productions in the above constructed grammar G_φ is polynomial in the number of clauses occurring in φ. As a consequence, the above reduction is polynomial time computable.

Proposition 11. *Let φ be a formula in 3-DNF with n variables and m clauses. Then $|G_\varphi| \leq 6m^2$, for $X \in \Gamma$, where G_φ is the grammar defined above which can be constructed in deterministic polynomial time.*

The construction of G_φ also fulfills the property that a word $w \in \{0,1\}^n$ is derivable in G_φ if and only if w—interpreted as a truth assignment—satisfies the formula φ, i.e., $L(G_\varphi) = \{ w \in \{0,1\}^n \mid w \models \varphi \}$. An immediate consequence of the following proposition is that G_φ generates all words of length n over $\{0,1\}$ if and only if φ is a tautology. In other words, the reduction from the tautology problem to the fixed length universality problem is *correct*.

Proposition 12. *Let φ be a formula in 3-DNF with n variables. Then, for all words $w \in \{0,1\}^n$, it holds that $w \in L(G_\varphi)$ if and only if $w \models \varphi$.*

The next step in our proof strategy is the construction of a grammar that generates the left quotient of a word with a finite language from a given grammar that generates this language. Before we can do this, we need two prerequisites. The first lemma states that any finite language can be generated by a grammar which has the property that the right-hand side of each of its productions is not longer than the length of a longest word in the language.

Lemma 13. *Let $X \in \Gamma$, G be an X-grammar generating a finite language, and $\ell := \max\{ |w| \mid w \in L(G) \}$. Then there is an X-grammar G' where all right-hand sides of productions are of length at most ℓ s.t. $L(G') = L(G)$ and $|G'| \leq |G|$.*

Next, a context-free grammar $G = (N, \Sigma, P, S)$ is said to be in *binary normal form* (2NF) if the right-hand side of all productions in P has length at most two, i.e., for all $A \to \alpha \in P$ it holds that $|\alpha| \leq 2$. Then the second lemma shows that this assumption does not constitute a major restriction.

Lemma 14. *Let $X \in \Gamma$ and G be an X-grammar generating a finite language. Assume that $\ell := \max\{\, |w| \mid w \in L(G)\,\}$. Then there is an X-grammar G' in binary normal form such that $L(G') = L(G)$ and $|G'| \leq |G| \cdot \ell$.*

Now that we have collected all necessary ingredients, we can finally prove the result on the left quotient of a word and a finite language. The proof uses both Lemmata 13 and 14 and a triple like construction from [11] that is similar to the well known triple construction for the intersection of a context-free language with a regular set. The theorem reads as follows:

Theorem 15. *Let $X \in \Gamma$ and G be an X-grammar generating a finite language whose longest word has length ℓ. Then, for every word $w \in \Sigma^*$, there is an X-grammar G' with $L(G') = w^{-1}L(G)$ and $|G'| \in O(|G| \cdot |w|^3 \cdot \ell)$.*

In the remainder of this section, we prove our main result on the inapproximability of grammatical descriptions of finite languages. The following result expresses upper and lower bounds on the X-complexity of the language L_φ, for $X \in \Gamma$. In the case of context-free grammars, a constant number of productions suffices to generate L_φ, however, when we turn to regular and linear grammars, the upper bound jumps to a linear number (w.r.t. the number of variables occurring in φ). On the other hand, if φ is not a tautology, then we obtain a lower bound on the X-complexity of L_φ, for $X \in \Gamma$, which is polynomial in the number of variables occurring in φ. The value c in the next lemma refers to the constant c used in the construction of L_φ.

Lemma 16. *Let $X \in \Gamma$ and let φ be a formula in 3-DNF over n variables. Then (1) $\mathsf{Xc}(L_\varphi) = O(n)$ if φ is a tautology and (2) $\mathsf{Xc}(L_\varphi) = \Omega(n^{c-4})$, otherwise.*

Now, we are ready to prove the main result of this paper:

Proof (of Theorem 10). In Proposition 11, we have obtained an upper bound on the number of productions in a grammar describing L_φ in terms of m, the number of clauses in φ. The gap-introducing part of our reduction is of course Lemma 16, which is formulated in terms of n, the number of variables in φ. In order to show our inapproximability result, we need to establish a relation between the number of clauses and variables in the formula φ. For instance, it is known that the 3-SAT problem remains NP-complete if we require that every variable occurs exactly four times [15]. This result of course implies coNP-completeness of the corresponding 3-DNF tautology problem, where the number of clauses is linear in the number of variables. So, without loss of generality, we will assume that the number of clauses in our instance of the 3-DNF tautology problem is linear in the number of variables. Now, we are finally in the position to fix the constant c, by choosing $c = 6$. Recall the definition of L_φ as

$$L_\varphi = L(G_\varphi) \cdot \{\&\} \cdot \{0, 1, \$, \#\}^{3c\cdot\lceil \log n\rceil + 2} \cup \{0, 1\}^n \cdot \{\&\} \cdot T_{c\cdot\lceil \log n\rceil}.$$

From Proposition 11, we deduce that for the grammar G_φ it holds that $|G_\varphi| = O(m^2) = O(n^2)$. When we combine the above upper bound with the upper bounds from Theorem 2 and the straightforward fact that $\mathsf{Xc}(\Sigma^\ell) \leq |\Sigma| \cdot \ell$ — using the bounds for union and concatenation (Theorems 3 and 6, respectively), we obtain that L_φ admits a regular grammar with p productions such that

$$p = O(n^2) + O(1) + O(\log n) + O(n) + O(1) + O(n^c) = O(n^6).$$

Towards a contradiction, assume that there is a polynomial time approximation algorithm A for the minimal number of productions problem within $o(p^{1/6})$. Then A could be used to decide in polynomial time whether φ is a tautology as follows: if φ is a tautology, then, by Lemma 16, $\mathsf{Xc}(L_\varphi) = O(n)$, for $X \in \Gamma$, otherwise, that is, if φ is not a tautology, then, by Lemma 16, we deduce that, for $X \in \Gamma$, it holds that $\mathsf{Xc}(L_\varphi) = \Omega(n^{c-4}) = \Omega(n^2)$. Consequently, the putative approximation algorithm A returns a grammar size of at most $o(p^{1/6}) \cdot O(n) = o(n^2)$ if and only if φ is a tautology. However, this solves the coNP-hard 3-DNF tautology problem in deterministic polynomial time, which implies $\mathsf{P} = \mathsf{NP}$. This shows that the X-complexity, for $X \in \Gamma$, of a given finite language cannot be approximated within a factor of $o(p^{1/6})$, unless $\mathsf{P} = \mathsf{NP}$. $\square$

6 The Smallest Grammar Problem for Finite Languages

The smallest grammar problem asks for the smallest context-free grammar that generates a *single* given word and its decision version has been shown to be NP-complete for unbounded [6] and fixed alphabets of size at least 24 [5]. In [6], it was also shown that the smallest grammar problem (w.r.t. unbounded alphabets) has an approximation ratio of at least $\frac{8569}{8568}$, unless $\mathsf{P} = \mathsf{NP}$. We will consider—w.r.t. fixed alphabets of size at least 5—another formulation of the smallest grammar problem that asks for the smallest grammar that generates a given *finite language* instead of just a single word. The authors in [6] defined the size of a grammar as the sum of the lengths of the right-hand sides of all productions. They slightly deviate from the classical definition from [12], which reads as follows: the *size* of an X-grammar $G = (N, \Sigma, P, S)$ is $|G|_s = \sum_{A \to \alpha \in P} (2 + |\alpha|)$. The *minimal X-size* of a finite language L is

$$\mathsf{Xs}(L) = \min\{\, |G|_s \mid G \text{ is an } X\text{-grammar with } L = L(G) \,\}.$$

The lower bound on the language T_n in terms of the X-complexity, for $X \in \Gamma$, immediately implies a lower bound of $\mathsf{Xs}(T_n) = \Omega(2^n)$ on the size of a minimal X-grammar generating T_n, since we can assume that the grammar is ε-free. This bound will suffice for our purpose. Along similar lines as in the proof of Lemma 16, we get a linear upper bound on the minimal grammar size for L_φ if φ is a tautology. However, the lower bound for the minimal grammar size of L_φ if φ is not a tautology asymptotically coincides with the one obtained for the minimal number of productions. Thus, Lemma 16, remains valid in case Xc is replaced by Xs. Then, we get an inapproximability result analogous to Theorem 10:

Theorem 17. *Let $X \in \Gamma$. Given an X-grammar of size s generating a finite language L, it is impossible to approximate $\mathsf{Xs}(L)$ within a factor of $o(s^{1/7})$, unless $\mathsf{P} = \mathsf{NP}$.*

Observe that our reduction scheme is robust enough to yield the same inapproximability result if we define the grammar size as in [6], i.e., as the sum of the right-hand sides of all productions. Thus, the result of Theorem 17 also holds for the alternative definition of grammar size.

References

1. B. Alspach, P. Eades, and G. Rose. A Lower-bound For the Number of Productions Required For A Certain Class of Languages. *Discrete Appl. Math.*, 6:109–115, 1983.
2. W. Bucher. A Note on a Problem in the Theory of Grammatical Complexity. *Theoret. Comput. Sci.*, 14(3):337–344, 1981.
3. W. Bucher, H. A. Maurer, and K. Culik II. Context-Free Complexity of Finite Languages. *Theoret. Comput. Sci.*, 28(3):277–285, 1983.
4. W. Bucher, H. A. Maurer, K. Culik II, and D. Wotschke. Concise Description of Finite Languages. *Theoret. Comput. Sci.*, 14(3):227–246, 1981.
5. K. Casel, H. Fernau, S. Gaspers, B. Gras, and M. L. Schmid. On the Complexity of Grammar-Based Compression over Fixed Alphabets. In I. Chatzigiannakis, M. Mitzenmacher, Y. Rabani, and D. Sangiorgi, editors, *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming*, volume 55 of *LIPIcs*, pages 122:1–122:14, Rome, Italy, 2016. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany.
6. M. Charikar, E. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, and S. Shelat. The Smallest Grammar Problem. *IEEE Trans. Inf. Theory.*, 51(7):2554–2576, 2005.
7. J. Dassow. Descriptional Complexity and Operations—Two Non-Classical Cases. In G. Pighizzini and C. Câmpeanu, editors, *Proceedings of the 19th International Workshop on Descriptional Complexity of Formal Systems*, number 10316 in LNCS, pages 33–44, Milano, Italy, 2017. Springer.
8. J. Dassow and R. Harbich. Production Complexity of Some Operations on Context-Free Languages. In M. Kutrib, N. Moreira, and R. Reis, editors, *Proceedings of the 14th Workshop on Descriptional Complexity of Formal Systems*, number 7386 in LNCS, pages 141–154, Braga, Portugal, 2012. Springer.
9. Sebastian Eberhard and Stefan Hetzl. On the compressibility of finite languages and formal proofs. *Information and Computation*, 259:191–213, 2018.
10. Y. Filmus. Lower bounds for context-free grammars. *Inform. Process. Lett.*, 111(18):895–898, 2011.
11. S. Ginsburg and E. H. Spanier. Quotients of Context-Free Languages. *J. ACM*, 10(4):487–492, 1963.
12. M. A. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, 1978.
13. M. Holzer and M. Kutrib. Descriptional Complexity—An Introductory Survey. In C. Martín-Vide, editor, *Scientific Applications of Language Methods*, pages 1–58. World Scientific, 2010.
14. H. B. Hunt, III. *On the time and tape complexity of languages.* Ph.D. thesis, Cornell University, Ithaca, New York, USA, 1973.
15. C. A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984.
16. Zs. Tuza. On the Context-Free Production Complexity of Finite Languages. *Discrete Appl. Math.*, 18(3):293–304, 1987.