# On Pumping Constants and Smallest Grammars for Context-Free Languages

Hermann Gruber[1] and Markus Holzer[2]

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

**Abstract.** We study the relationship between the minimal context-free pumping constant of a context-free language and the size of the context-free grammar that generates it. For the size, we consider the sum of the lengths of the right-hand sides of the productions and the total number of symbols to write the productions, including the "$\rightarrow$" symbol in each production; the latter size concept is known in the literature as symbol complexity. We prove tight bounds for both size concepts. Furthermore, we apply our results to some open problems on the symbol complexity of languages. In particular, we show that for the language $L_n = \{a^n\}$ the symbol complexity is at least $6\log_4 n$ and at most $6\log_4 n + O(\frac{\log n}{\log\log n})$.

## 1 Introduction

Let $G = (N, T, P, S)$ be a *context-free grammar* (CFG), where $N$ and $T$ are disjoint alphabets of *nonterminals* and *terminals*, respectively, $S \in N$ is the *axiom*, and $P$ is a finite set of *productions* of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in (N \cup T)^*$. As usual, the transitive closure of the derivation relation $\Rightarrow_G$ is written as $\Rightarrow_G^*$. If there is no danger of confusion, we simply write $\Rightarrow$ ($\Rightarrow^*$, respectively) instead of $\Rightarrow_G$ ($\Rightarrow_G^*$, respectively). The *language generated* by $G$ is defined as

$$L(G) = \{\, w \in T^* \mid S \Rightarrow_G^* w \,\}.$$

For every context-free language the Bar-Hillel lemma [1] applies, which reads as follows:

**Lemma 1.** *Let $L$ be a context-free language over $\Sigma$. Then, there is a constant $p$ (depending on $L$) such that the following holds: If $z \in L$ and $|z| \geq p$, then there are words $u, v, w, x, y \in \Sigma^*$ such that $z = uvwxy$, $|vx| \geq 1$, $|vwx| \leq p$, and $uv^twx^ty \in L$ for $t \geq 0$—it is then said that $v$ and $x$ can be (simultaneously)* pumped *in $z$.*

For a context-free language $L$, let $\texttt{mpcf}(L)$ denote the minimal number $p$ satisfying the conditions of Lemma 1. We find the following situation for context-free languages, which follows from the proof of the pumping lemma given in [11, Chapter 6]:

**Theorem 2.** *Let $L$ be generated by the context-free grammar $G = (N, T, P, S)$. Then*

$$\mathit{mpcf}(L) \leq m^{2n+3},$$

*where $n := |N|$ and $m := \max\{\, 2, |\alpha| \mid A \to \alpha \in P \,\}$.*

If the given context-free grammar is in Chomsky normal form,[3] the proof in [13] yields the bound $\mathtt{mpcf}(L) \leq 2^n$. Notice, however, that the conversion to normal form incurs a size blow-up in the worst case [14].

When considering linear context-free languages, the bound on the minimal pumping constant becomes linear as we will see next. 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 $\alpha \in T^*(N \cup \{\varepsilon\})T^*$—here $\varepsilon$ refers to the *empty word.* The pumping lemma for linear-context free languages reads like the pumping lemma for context-free languages, but with one exception: instead of $|vwx| \leq p$, now the condition $|uvxy| \leq p$ is required—see [13, page 143, Exercise 6.11]. For a linear context-free language $L$ let $\mathtt{mplin}(L)$ denote the minimal number $p$ satisfying the conditions of the pumping lemma for linear context-free languages. The next result was shown in [7, Thm. 6]:

**Theorem 3.** *Let $L$ be a* linear *context-free language generated by the linear context-free grammar $G = (N, T, P, S)$. Then*

$$\mathit{mplin}(L) \leq (m - 1) \cdot n + 2,$$

*where $n := |N|$ and $m := \max\{\, |\alpha| \mid A \to \alpha \in P \,\}$.*

In the above theorems, the number of nonterminals and the maximal length of right-hand sides of the productions play a crucial *rôle.* Are there any other relations between the minimal pumping constant of a context-free language and some descriptional complexity measure of context-free languages that is closer to the size of the grammar that generates the language in question? We partially answer this question in the forthcoming.

## 2 Results on Minimal Context-Free Pumping Constants

Several concurrent notions to measure the actual *size* of a context-free grammar have been proposed in the literature, see, e.g., [2,6,8,11,12,15]. The most intuitive ways to measure the size of a context-free grammar, as they appear in textbooks on automata theory, are

1. to count the sum of the lengths of the right-hand sides of the rules, or
2. to count the total number of symbols to write down the rules, including the "$\to$" symbol in each production.

[3] A context-free grammar $G = (N, T, P, S)$ is in *Chomsky normal form* if every production is either of the form $A \to a$ or $A \to BC$ or $S \to \varepsilon$, for $A, B, C \in N$ and $a \in T$.

The latter way to measure the size of a context-free grammar facilitates telling apart the start and end of each production, and, originating from [8]. This seems to be the oldest explicit notion of grammar size. The recent literature in the field of descriptional complexity [3,10] refers to this measure as the *symbol complexity* of a context-free grammar. The convention to use the sum of the length of the right-hand sides of the productions has been adopted by papers dealing with the *smallest grammar problem*. In that optimization problem, the task is, given word $w$, to find a context-free grammar $G$, such that (i) the grammar $G$ generates the single word $w$ and (ii) the size of $G$ is as small as possible. The first systematic study of grammar-based compression [2] proved that the size of $G$ is always in $\Omega(\log |w|)$; our results below can be seen as an extension of that result to general context-free grammars.

We start our investigation by determining a tight bound on the minimal context-free pumping constant in terms of the sum of the lengths of the right-hand sides of the grammar. For this, we need some properties of a function $\rho$ which became known as the maximum product function w.r.t. a partition of the number [9], that is, the value $\rho(n)$ is obtained by maximizing the product $\prod_{i=0}^{h} k_i$ of positive integers $k_i$ subject to the condition $\sum_{i=0}^{h} k_i = n$, for $1 \leq k_i \leq n$ with $0 \leq i \leq h$. Moreover, $\rho(n)$ obeys a recursive relation, namely, $\rho(1) = 1$, $\rho(2) = 2$, $\rho(3) = 3$, $\rho(4) = 4$, and $\rho(n) = 3 \cdot \rho(n-3)$, for $n > 4$. For further uses of this function, see sequence A000792 (with offset 1) in "The On-line Encyclopedia of Integer Sequences" (OEIS)—Table 1. This function became also known in graph

| | OEIS | Sequence |
|---|---|---|
| P. Halmos | A000792 (offset 1) | 1, 2, 3, 4, 6, 9, 12, 18, 27, 36, 54, 81, 108, 162, 243, 324, 486, 729, 972, 1458, ... |
| J. Derbyshire | A193286 | 1, 2, 3, 4, 5, 6, 7, 9, 12, 16, 20, 25, 30, 36, 48, 64, 80, 100, 125, 150, 192, 256, 320, 400, 500, 625, 768, 1024, 1280, 1600, 2000, 2500, 3125, ... |

**Table 1.** OEIS sequences.

theory as the Moon-Moser bound [18], see also [17]. For the proofs to come, we define $\rho(0) = 0$, which is different from the convention used for A000792. Then the following statement on the $\rho$ function is immediate: (i) the recurrence is a literal translation of the definition of the $\rho$ function as the maximum value of the product subject to the sum condition, and (ii) the closed formula follows since one takes as many of the $k_i$'s as possible to be 3 and then use one or two 2's only; larger $k_i$'s with $k_i \geq 4$ can be replaced by 2 and $k_i - 2$, since $2(k_i - 2) \geq k_i$ for $k_i \geq 4$, in order to increase the product value. Thus, we obtain the following result.

**Lemma 4.** *For a positive integer $n$, let the function $\rho(n)$ be given by the recurrence $\rho(i) = i$ for $0 \leq i \leq 3$, and*

$$\rho(n) = \max_{1 \leq k < n} \{k \cdot \rho(n-k)\} \quad \textit{for } n \geq 4.$$

*Then*

$$\rho(n) = \begin{cases} 0, & \textit{for } n = 0 \\ 1, & \textit{for } n = 1 \\ 3^{\frac{n}{3}}, & \textit{for } n \equiv 0 \pmod 3 \textit{ and } n \geq 1 \\ 4 \cdot 3^{\frac{n-4}{3}}, & \textit{for } n \equiv 1 \pmod 3 \textit{ and } n \geq 4 \\ 2 \cdot 3^{\frac{n-2}{3}}, & \textit{for } n \equiv 2 \pmod 3). \end{cases}$$

*Also, $\rho(n-k)+k \leq \rho(n)$, for $n \geq 1$, and a positive integer $k$ with $1 \leq k \leq n$.* □

Now we are ready for the tight upper bound on the minimal pumping constant of context-free languages in terms of the sum of the lengths of the right-hand sides of the grammar.

**Theorem 5.** *Let $L$ be generated by the context-free grammar $G = (N, T, P, S)$, where $L$ is not empty. Then*

$$\textit{mpcf}(L) \leq \rho(m_G) + 1,$$

*where $m_G := \sum_{(A \to \alpha) \in P} |\alpha|$ and $\rho$ is the function from Lemma 4. Furthermore, for every $m \geq 0$, there exists a context-free grammar $G_m$ witnessing that this bound is tight.*

*Proof.* We prove the upper bound using the following statement: Let $w$ be a word in $L(G)$; if $w$ cannot be pumped, then $|w| \leq \rho(m_G)$.

We prove this statement by lexicographic induction on the the parameter $m_G$ and the minimum depth of a parse tree $T$ in $G$ for $w$. If $d(T)$ denotes the depth of parse tree $T$, then for the base case of the induction we have a grammar $G$ with $m_G = 0$, and a parse tree of depth 1. In the latter, the root is labeled by the start symbol $S$, and there are $|w| = m_G$ terminal leaves. We obtain $\rho(0) = 0$, thus establishing the base case of the induction.

For the induction step, assume that the claim holds for all grammars $G'$ with $m_{G'} < m_G$ and, in $G$, for all words admitting a parse tree of depth at most $d-1$. We consider two cases. Consider a parse tree $T$ of depth $d$ for $w$:

**Case 1.** There is a path in $T$ from the root to some leaf labeled with a terminal symbol, such that the start symbol $S$ appears at least twice.
The start symbol appears as a label at the root of the parse tree. There is also a proper subtree of $T$ of depth at most $d-1$, whose root is labeled by $S$. This subtree is parse tree of a word $y$ according to the grammar $G$, and $y$ is a contiguous subword of $w$.

**Case 1a.** Assume $y = w$. Then we have a contradiction, since we chose $w$ as a word not admitting a parse tree of depth less than $d$.

**Case 1b.** Otherwise, assume $y \neq w$. Then we can write $w = xyr$ with $|x| > 0$ or $|y| > 0$, and $S \Rightarrow^* xSz$ as well as $S \Rightarrow^* y$. Thus, we have $x^i y z^i \in L(G)$, for all $i > 0$, and $w$ can be pumped. Similarly, we have a contradiction also in this subcase.

**Case 2.** On every path in $T$ from the root to some leaf labeled with a terminal symbol, the start symbol $S$ appears only once.

There is a production $p = (S \to \beta)$ that corresponds to the root of the parse tree and its children.

**Case 2a.** Right-hand side $\beta$ contains at least one terminal symbol. Let $p'$ denote the production obtained from $p$ by deleting the first terminal symbol $t$ on its right-hand side, and let $G'$ denote the grammar obtained from $G$ by replacing the production $p$ with $p'$. Then, $m_{G'} = m_G - 1$, and there is a word $w'$ such that $w'$ is obtained from $w$ by deleting a single letter, and $w'$ admits a parse tree of depth $d$ according to $G'$. By the induction assumption, $|w| = |w'| + 1 \leq \rho(m_G - 1) + 1$. Recall from Lemma 4 that we have $\rho(m_G - 1) + 1 \leq \rho(m_G)$, thus establishing the bound in this subcase.

**Case 2b.** Right-hand side $\beta$ contains only nonterminal symbols. If there are $j$ occurrences of nonterminals in $\beta$, then there are $j$ subtrees of depth at most $d-1$ whose leaves spell out a subword of $w$ each. Together these subwords make up the word $w$. Notice that neither of the subtrees uses the variable $S$, and also not $p$. If $B$ is the nonterminal label of such a subtree, it is a parse tree of the context-free grammar

$$G_B = (N \setminus \{S\}, T, P \setminus \{p\}, B).$$

We next show that we can assume $1 \leq j < m_G$: because $|\beta| = j$ and $|\beta| \leq m_G$, we have $m_G - j \geq 0$. Also, if $j = m_G$, then no terminals can appear on the right side of any production. It follows that $|w| = 0$. Since $d(T) \geq 2$, we must have $j \geq 1$. Thus, $1 \leq j < m_G$ if $w$ is not empty.

By the induction assumption, each of the $j$ occurrences of a nonterminal generates a contiguous substring of $w$, and each such substring has length at most $\rho(m_G - j)$. Hence we obtain

$$|w| \leq \sum_{i=1}^{j} \rho(m_G - j), \text{ for some } j \text{ with } 1 \leq j < m_G.$$

Observe, that all summands in the above sum are identical, so we can rewrite the inequality as $|w| \leq j \cdot \rho(m_G - j)$. In turn, by Lemma 4, we have $j \cdot \rho(m_G - j) \leq \rho(m_G)$ for each such $j$. Combining the last two inequalities yields $|w| \leq \rho(m_G)$ also in this case, and the proof of the upper estimate is completed.

Regarding the grammars witnessing that the bound is tight, let us first consider an example with $m = 13$. With $n = \lfloor \frac{m}{3} \rfloor + 1 = 5$, grammar $G_{13}$ is given with variables $A_1, A_2, \dots A_5$, alphabet $\Sigma = \{a\}$, and productions

$$\begin{aligned} A_i &\to A_{i+1}A_{i+1}A_{i+1}, \quad \text{for } 1 \leq i \leq 3,\\ A_4 &\to A_5A_5, \end{aligned}$$

and

$$A_5 \to aa,$$

and the axiom $A_1$. The pattern generalizes as follows: grammar $G_m$ has $n$ variables, where $n = \frac{m}{3}$ when $m$ is divisible by 3, and $n = \lfloor \frac{m}{3} \rfloor + 1$ otherwise; of these variables, all but the last two produce 3 symbols each. When $m \equiv 1 \pmod 3$ and $m \geq 4$, the last two variables produce only 2 symbols each; when $m \equiv 2 \pmod 3$, then the penultimate variable produces 3 symbols, and the last variable produces 2 symbols; when $m \equiv 0 \mod 3$, then also the last two variables produce 3 symbols each. For the edge case $m = 1$, we have a single production that produces a single terminal symbol. It is readily seen that $G_m$ generates a single unary word of length $\rho(m)$. $\square$

Let us quickly consider the case of linear context-free languages. Regarding the sum of the lengths of the right-hand sides of the grammar, we can proceed as in the proof of Theorem 5, but the analysis degenerates to some easy cases. The proof is left as an exercise to the reader; we also note that the new bound implies the previous one from [7, Thm. 6].

**Theorem 6.** *Let the language $L$ be generated by the linear context-free grammar $G = (N, T, P, S)$, where $L$ is not empty. Then*

$$\mathit{mplin}(L) \leq m_G + 1,$$

*where $m_G := \sum_{(A\to\alpha)\in P} |\alpha|$. Furthermore, for every $m \geq 1$, there exists a linear context-free grammar $G_m$ witnessing that this bound is tight.* $\square$

Next let us consider the *symbol complexity* of a grammar, which counts the total number of symbols to write down the rules, including the "$\to$" symbol in each production. As before, we first state a technical lemma, which relates to an integer sequence. The latter is the solution to a recreational mathematical puzzle, described in the following. A text editing software allows for the following "keystroke" actions:

– typing a single character,
– selecting all text (Ctrl+A on Windows and Linux operating systems, Cmd+A for Mac users),
– copying the selection to the clipboard (Ctrl+C on Windows and Linux operating systems, Cmd+C for Mac users), and
– pasting the clipboard contents (Ctrl+V on Windows and Linux operating systems, Cmd+V for Mac users).

The keyboard problem asks, given a positive integer $n$, for the maximum amount of characters that can be produced using $n$ keystroke actions. Note that the copy command de-selects the buffer text, i.e., we perform copying with replacement, which means that the first paste, or simple insertion, after copying, eliminates, overwrites, or otherwise renders obsolete the currently existing text output. For instance, the 11-keystroke sequence $aaaACVVACVa$ (simple character $a$ and select, copy, and paste simplified) outputs 7 characters—the buffer change is depicted by the sequence

$$\begin{aligned}\varepsilon \to^a a \to^a aa \to^a aaa \to^A aaa \to^C aaa \to^V aaa \to^V aaa\,aaa\\ \to^A aaa\,aaa \to^C aaa\,aaa \to^V aaa\,aaa \to^a aaa\,aaa\,a.\end{aligned}$$

For 11 keystrokes the maximum amount of characters obtainable is 20—by the keystroke sequence $aaaaaACVVVV$. A solution of the problem can be found in [19], and is named $\sigma(n)$—see Table 1 and for more information consult A193286 in OEIS. The value $\sigma(n)$ is obtained by maximizing the product $\prod_{i=0}^{h} k_i$ of positive integers $k_i$ subject to the condition $\sum_{i=0}^{h} k_i = n - 2h$, where $h$ is the number of copy operations. Moreover, $\sigma(n)$ obeys a recursive relation: we have $\sigma(n) = n$ for $1 \leq n \leq 7$; a few sporadic values occur with $\sigma(8) = 3^2$, $\sigma(12) = 5^2$, $\sigma(13) = 6 \cdot 5$, $\sigma(19) = 5^3$, $\sigma(20) = 6 \cdot 5^2$, $\sigma(26) = 5^4$, and $\sigma(33) = 5^5$. For all other $n$, the formula $\sigma(n) = 4 \cdot \sigma(n-6)$ applies; see [19] for further background and explanation. We obtain the following result (again, we define that $\sigma(0) = 0$):

**Lemma 7.** *For a positive integer $n$, let the function $\sigma(n)$ be given by the recurrence $\sigma(n) = n$, for $0 \leq n \leq 7$, and*

$$\sigma(n) = \max_{1 \leq k < n-2} \{k \cdot \sigma(n-k-2)\}, \quad \textit{for } n \geq 8.$$

*Then*

$$\sigma(n) = \begin{cases} 5^2 \cdot 4^{(n-12)/6}, & \textit{for } n \bmod 6 \equiv 0 \textit{ and } n \geq 12\\ 5^3 \cdot 4^{(n-19)/6}, & \textit{for } n \bmod 6 \equiv 1 \textit{ and } n \geq 19\\ 5^4 \cdot 4^{(n-26)/6}, & \textit{for } n \bmod 6 \equiv 2 \textit{ and } n \geq 26\\ 5^5 \cdot 4^{(n-33)/6}, & \textit{for } n \bmod 6 \equiv 3 \textit{ and } n \geq 33\\ 4^{(n+2)/6}, & \textit{for } n \bmod 6 \equiv 4\\ 5 \cdot 4^{(n-5)/6}, & \textit{for } n \bmod 6 \equiv 5. \end{cases}$$

*The finite number of remaining cases for $\sigma(n)$ can be found in Table 1 under A193286.* □

*Proof.* The recurrence on $\sigma(n)$ is a literal translation of its definition as the maximum product $\prod_{i=0}^{h} k_i$ of positive integers $k_i$ subject to the condition $\sum_{i=0}^{h} k_i = n - 2h$, where $h$ is the number of multiplications used.

Provided $n$ is large enough, we immediately obtain a formula for $\sigma(n)$ for each remainder of $n$ modulo 6 from the original recurrence $\sigma(n) = 4 \cdot \sigma(n-6)$, together with the sporadic values given above. The tedious details are left to the reader. □

Next we study a recurrence, which looks very similar to the recurrence of $\sigma$, with a few slight exceptions:

**Theorem 8.** *For an integer $n \geq 2$, let the function $\mu(n)$ be given by the recurrence $\mu(2) = 0$, $\mu(3) = 1$, $\mu(4) = 2$, $\mu(5) = 3$, and*

$$\mu(n) = \max\{\, \mu(n-1) + 1, \max_{2 \leq k \leq n-4} \{k \cdot \mu(n-k-2)\}\,\}, \text{ for } n \geq 6.$$

*Then $\mu(n+2) = \sigma(n)$, for $n \geq 0$. In particular, we have*

$$\mu(n) = \begin{cases} 5^0 \cdot 4^{n/6}, & \text{for } n \bmod 6 \equiv 0 \text{ and } n \geq 6 \\ 5^1 \cdot 4^{(n-7)/6}, & \text{for } n \bmod 6 \equiv 1 \text{ and } n \geq 7, \\ 5^2 \cdot 4^{(n-14)/6}, & \text{for } n \bmod 6 \equiv 2 \text{ and } n \geq 14 \\ 5^3 \cdot 4^{(n-21)/6}, & \text{for } n \bmod 6 \equiv 3 \text{ and } n \geq 21 \\ 5^4 \cdot 4^{(n-28)/6}, & \text{for } n \bmod 6 \equiv 4 \text{ and } n \geq 28 \\ 5^5 \cdot 4^{(n-35)/6}, & \text{for } n \bmod 6 \equiv 5 \text{ and } n \geq 35. \end{cases}$$

*The finite number of remaining cases for $\mu(n)$ can be found in Table 1 under A193286, taking care of the offset by two.*

*Proof.* First, reconsider the recurrence for the $\sigma$ function as stated in Lemma 7. One observes that $\sigma(n-1)+1 \leq \sigma(n)$, for $n \geq 2$. This is due to the fact that $\sigma(n)$ grows exponentially for $n \geq 33$ and the remaining finite number of cases can be verified by inspecting the appropriate sequence given in Table 1. Thus, for $n \geq 8$, we can safely replace the $\sigma$-recurrence with

$$\sigma(n) = \max\{\, \sigma(n-1) + 1, \max_{1 \leq k < n-2} \{k \cdot \sigma(n-k-2)\}\,\}.$$

Now we are ready to show that $\mu(n+2) = \sigma(n)$, for $n \geq 0$ by induction on $n$. Easy calculations show that $\mu(n+2) = \sigma(n)$, for $0 \leq n \leq 7$; these are left to the reader. Then, for the induction step, let $n \geq 8$. For $\mu$, the recurrence applies, namely

$$\begin{aligned} \mu(n+2) &= \max\{\, \mu((n+2)-1) + 1, \max_{2 \leq k \leq (n+2)-4} \{k \cdot \mu((n+2)-k-2)\}\,\} \\ &= \max\{\, \mu(n+1) + 1, \max_{2 \leq k \leq n-2} \{k \cdot \mu(n-k)\}\,\}. \end{aligned}$$

For the inner maximum, we first show that the range for the variable $k$ can be extended to include $k = 1$ without altering the value of the maximum: By induction, we know that $\mu$ is strictly monotonically increasing for values up to $n-1$, because of the corresponding property of $\sigma$. Thus,

$$2 \cdot \mu(n-2) \geq 2 \cdot (\mu(n-1) + 1) > 1 \cdot \mu(n-1),$$

and hence

$$\max_{2 \leq k \leq n-2} \{k \cdot \mu(n-k)\} = \max_{1 \leq k \leq n-2} \{k \cdot \mu(n-k)\}.$$

The term $\max_{1\leq k\leq n-2}\{k\cdot\mu(n-k)\}$, when spelled out, reads as

$$\max\{\,1\cdot\mu(n-1), 2\cdot\mu(n-2), \ldots, (n-3)\cdot\mu(3), (n-2)\cdot\mu(2)\,\}.$$

Since $\mu(2)=0$ by definition, this simplifies to $\max_{1\leq k\leq n-3}\{k\cdot\mu(n-k)\}$. Then, by the induction hypothesis, the $\mu$-terms can be replaced by appropriate $\sigma$-terms. Hence, the term is $\max_{1\leq k\leq n-3}\{k\cdot\sigma(n-k-2)\}$. Next, we combine this term with the outer maximum, which means that

$$\begin{aligned}\mu(n+2) &= \max\{\,\mu(n+1)+1, \max_{1\leq k\leq n-2}\{k\cdot\mu(n-k)\}\,\}\\ &= \max\{\,\sigma(n-1)+1, \max_{1\leq k\leq n-3}\{k\cdot\sigma(n-k-2)\}\,\}\\ &= \sigma(n),\end{aligned}$$

where we have applied the alternative recurrence for $\sigma$ with two nested maximum operations. This completes the induction proof.

To finish the proof of the theorem, evaluating the formula for $\sigma$ from Lemma 7 at $n-2$ somewhat magically reveals a pattern:

$$\mu(n)=\sigma(n-2)=\begin{cases}5^0\cdot 4^{n/6}, & \text{for } n \bmod 6\equiv 0\\ 5^1\cdot 4^{(n-7)/6}, & \text{for } n \bmod 6\equiv 1 \text{ and } n\geq 7,\\ 5^2\cdot 4^{(n-14)/6}, & \text{for } n \bmod 6\equiv 2 \text{ and } n\geq 14\\ 5^3\cdot 4^{(n-21)/6}, & \text{for } n \bmod 6\equiv 3 \text{ and } n\geq 21\\ 5^4\cdot 4^{(n-28)/6}, & \text{for } n \bmod 6\equiv 4 \text{ and } n\geq 28\\ 5^5\cdot 4^{(n-35)/6}, & \text{for } n \bmod 6\equiv 5 \text{ and } n\geq 35,\end{cases}$$

and the proof is completed. □

Observe, that the pattern mentioned above simplifies to

$$\mu(n)=5^{n \bmod 6}\cdot 4^{(n-n_0)/6} \text{ for all } n\geq n_0, \text{ with } n_0 = 7\cdot(n \bmod 6) \qquad (1)$$

and will be later used in order to obtain a lower bound on the symbol complexity of the language $L_n=\{a^n\}$ in Theorem 13.

Now, we are ready to estimate the minimal context-free pumping constant in terms of the symbol complexity of the underlying context-free grammar.

**Theorem 9.** *Let $L$ be generated by the context-free grammar $G=(N,T,P,S)$, where $L$ is not empty. Then*

$$\mathit{mpcf}(L)\leq\mu(s_G)+1,$$

*where $s_G := \sum_{(A\to\alpha)\in P}(2+|\alpha|)$ and $\mu$ is the function from Theorem 8. Furthermore, for every $s\geq 2$, there exists a context-free grammar $G_s$ witnessing that this bound is tight.*

*Proof.* For the upper bound, the proof runs essentially along the same lines as the proof of Theorem 5, but instead of the parameter $m_G$, we express the bound in terms of $s_G$.

In **Case 2b** of the proof, some care has to be taken. If the right-hand side $\beta$ of the production $p = (S \to \beta)$ consists of a single variable $A$, then let $G'$ denote the grammar obtained from $G$ by removing $p$ and using $A$ as the start symbol. The grammar $G'$ has $s_G - 3$ symbols, and generates the same word $w$ with a parse tree of lower depth.

So, the interesting subcase of **Case 2b** that remains is $|\beta| \geq 2$. Here, we estimate $|\beta| \leq s_G - 4$, taking the arrow symbol and the left-hand side of the production $p$ into account, as well as the fact that, in **Case 2b**, there is at least one other production different from $p$. We note for later reference that estimate $2 \leq |\beta| \leq s_G - 4$ implies that this subcase can only appear for $s_G \geq 6$. Also, the production $p$ accounts for $|\beta| + 2$ symbols, so the recurrent bound changes to $|w| \leq j \cdot \sigma(s_G - 2 - j)$, with $2 \leq j \leq s_G - 4$. Altogether, we obtain the following recurrence, with start value $\mu(2) = 0$:

$$\mu(s) = \begin{cases} 0, & \text{if } s = 2 \\ \mu(s-1) + 1, & \text{if } 3 \leq s \leq 5, \\ \max\{\, \mu(s-1) + 1, \max\limits_{2 \leq j \leq s-4} \{j \cdot \mu(s-2-j)\}\,\}, & \text{if } s \geq 6, \end{cases}$$

obtaining the recurrence already studied in Theorem 8.

For the lower bound, we can construct a context-free grammar $G_s$ generating a single unary word, where the lengths of the right-hand sides are as in the factorization of $\sigma(s)$ given in the formula. □

For linear context-free grammars, the bound is again very simple—the tight bound is attained by grammars with a single production. The straightforward proof is omitted.

**Theorem 10.** *Let the language $L$ be generated by the linear context-free grammar $G = (N, T, P, S)$, where $L$ is not empty. Then*

$$\mathtt{mplin}(L) \leq s_G - 1,$$

*where $s_G := \sum_{(A \to \alpha) \in P} (2 + |\alpha|)$. Furthermore, for every $s \geq 2$, there exists a linear context-free grammar $G_s$ witnessing that this bound is tight.* □

## 3 More on the Symbol Complexity of Languages and Operations

In this section, we apply these new insights to questions regarding descriptional complexity of context-free languages and language operations. In [3], results are presented which show how the required number of variables, productions and symbols can behave under the operations union, concatenation and star. The recurring question is the following: what is the range of complexities that can be attained by applying a language operation to languages of complexity $m$ and $n$? We make this more precise for the symbol complexity under the concatenation operation. Following [3] define

$$g_{\cdot}^{\mathtt{symb}}(m,n) = \{\, \mathtt{symb}(L_1 \cdot L_2) \mid \mathtt{symb}(L_1) = m \text{ and } \mathtt{symb}(L_2) = n \,\},$$

where $\mathtt{symb}(L)$ refers to the symbol complexity of $L$, which is the minimum value among the symbol complexity of all context-free grammars generating $L$. Note that $g_{\cdot}^{\mathtt{symb}}(m,n) = g_{\cdot}^{\mathtt{symb}}(n,m)$ for all $m$ and $n$.

Compared to the number of variables [5] or productions [4], the symbol complexity of context-free languages seems to be more difficult to tackle. For example, although it is known that all languages of symbol complexity 3 consist of a single word of length 1, there are gaps in the known values for $g_{\cdot}^{\mathtt{symb}}(3,n)$, see [10, pp. 106f.]—see Table 2.

| $n$ | 3 | 4 | 5 | 6 | $\geq 7$ |
|---|---|---|---|---|---|
| $g_{\cdot}^{\mathtt{symb}}(3,n)$ | $\{4\}$ | $\{5\}$ | $\{6,7\}$ | $7 \in \cdot$ $\cdot \not\ni 0,1,\ldots,6,14,15,\ldots$ | $\cdot \not\ni 0,1,\ldots,6,n+8,n+9,\ldots$ |

**Table 2.** Symbol complexity of concatenation.

Let $L_n = \{a^n\}$, for $n \geq 0$. With the help of this language we can prove the following result:

**Theorem 11.** *For $n \geq 2$, we have $g_{\cdot}^{\mathit{symb}}(3,n) \ni n+1$.*

*Proof.* The cases $n \leq 6$ are treated in [10], so we assume $n \geq 7$. Consider the unary singleton languages $L_k = \{a^k\}$, for $k \geq 1$. Then by Theorem 9, we have $\mathtt{symb}(L_{\mu(n)}) = n$, and in particular $\mathtt{symb}(L_1) = 3$. Again by Theorem 9, the concatenation of these two languages cannot be generated with $n$ symbols. On the other hand, as observed in [8], we have $\mathtt{symb}(L_{k+1}) \leq \mathtt{symb}(L_k) + 1$, for all $k \geq 0$, thus $\mathtt{symb}(L_{\mu(n)+1}) = n+1$, as desired. $\square$

With the knowledge of $\mathtt{symb}(L_{\mu(n)})$ and $\mathtt{symb}(L_{\mu(n)+1})$, one can sporadically fill other gaps in the known values of $g_{\cdot}^{\mathtt{symb}}(m,n)$—in particular when $m$ and $n$ are small.

**Theorem 12.** *For $m,n \geq 3$ with $m+n \leq 12$, we have $g_{\cdot}^{\mathit{symb}}(m,n) \ni m+n-2$.*

*Proof.* For $1 \leq k \leq 7$, we have $\mathtt{symb}(L_k) = k+2$. For $k = 8$, we have $\mu(9)+1 = 8$, which implies that $\mathtt{symb}(L_k) = k+2$ also in that case. Thus, we can use $L_{m-2}$ and $L_{n-2}$ as witness languages—their concatenation $L_{m+n-4}$ requires $m+n-2$ symbols. $\square$

Only one of the values in the above theorem, namely $g^{\texttt{symb}}(4,5) \ni 7$, had been previously determined in [10]. To give a final example, we evaluate $\texttt{symb}(L_4) = 6$, $\texttt{symb}(L_8) = \texttt{symb}(L_9) = 10$, $\texttt{symb}(L_{12}) = 11$ and $\texttt{symb}(L_{13}) = 12$, to obtain $g^{\texttt{symb}}(6,10) \supseteq \{11,12\}$.

In the remainder of this section we are interested in the question, what is the symbol complexity of $L_n = \{a^n\}$ in general? An exact characterization may be out of reach. In [2], a clever construction is given, which, among other ingredients, works by recursively tripling the word length. However, in that paper, the goal is to minimize the sum of lengths of the right-hand sides in the grammar. For symbol complexity, the strategy needs to be adapted, so that it is based on quadrupling.

**Theorem 13.** *Let $n \geq 1$ be an integer and $L_n = \{a^n\}$. Then*

$$6\log_4 n \leq \texttt{symb}(L_n) \leq 6\log_4 n + O\left(\frac{\log n}{\log\log n}\right).$$

*Proof.* The lower bound can be derived from Equation 1 and Theorem 9 for large enough $s$: the minimum size is attained when $\mu(s) = n$ for some $s$. Then

$$n = 5^{s \bmod 6} \cdot 4^{(s-s_0)/6} \text{ with } s_0 = 7 \cdot (s \bmod 6),$$

for large enough $s$. By applying the logarithm in base 4 to the above formula, with $r = s \bmod 6$, we obtain:

$$\log_4(n) = (\log_4 5)\cdot r + \frac{s-7r}{6}.$$

Multiplying by six and solving for $s$ results in

$$s = 6\log_4(n) - (6\log_4 5)\cdot r + 7r = 6\log_4(n) + (7 - \underbrace{6\log_4 5}_{\approx 6.958})\cdot r.$$

The right-hand side is minimal when $r = 0$, so $s \geq 6\log_4(n)$, which is the desired lower bound.

For the upper bound, the algorithm is analogous to the proof of [2, Thm. 11]. Let $b$ be a power 4, that is, $b = 4^j$, to be fixed later. The idea is to represent $n$ in base $b$ as $n = \sum_{i=0}^{t} d_i b^i$, or merely, using Horner's rule,

$$n = (((d_t b + d_{t-1})b + d_{t_2})\, b + \cdots + d_1)\, b + d_0.$$

Here, $t = \lfloor \log_b n \rfloor$.

First, we introduce nonterminals $T_1, T_2, \ldots, T_{b-1}$ with productions $T_1 \to a$ and $T_{i+1} \to aT_i$, for $1 \leq i < b-1$, so we can generate strings of each short length at the cost of at most $4b$ symbols, without the need for extra symbols in the construction to come.

We prove by induction on $n$ that for $n \geq 1$, we can generate $a^n$ with at most $(6j+8)t$ additional symbols—in addition to those productions generating short strings, which we just have introduced. The cases with $n \leq b-1$ are readily verified, since $t = 0$ then, and we need no additional symbols in that case.

Now, assume the claim holds for all integers up to $n-1$. Let $q = \lfloor n/b \rfloor$ and $r = n \bmod b$, then $n = bq + r$ and $t - 1 = \lfloor \log_b q \rfloor$. Using the inductive hypothesis, there is a grammar $G_q$ generating $a^q$ with $(6j+8)(t-1)$ additional symbols. Let $G_q = (N_q, \{a\}, P_q, S_q)$, where the set of nonterminals $N_q$ contains already $T_1, T_2, \ldots, T_{b-1}$ and the set of productions $P_q$ the corresponding $T_i$-rules as described above.

We construct the context-free grammar $G_n = (N_n, \{a\}, P_n, S_n)$ with

$$N_n = N_q \cup \{X_1, X_2, \ldots, X_j\} \cup \{S_n\},$$

where $X_1, X_2, \ldots, X_j$ and $S_n$ are new pairwise different nonterminals not contained in $N_q$. Moreover $P_n$ contains all rules from $P_q$ and in addition

$$\begin{aligned} &S_n \to X_j T_r, \\ &X_i \to X_{i-1}X_{i-1}X_{i-1}X_{i-1}, \text{ for } 2 \leq i \leq j \end{aligned}$$

as well as

$$X_1 \to S_q.$$

Then, the variable $X_j$ generates a unary word of length $q \cdot 4^j$, and the nonterminal $S_n$ is responsible for adding length $r$ to this. In the induction step, we introduced at most $4 + 6j + 4 = 6j + 8$ new symbols, as desired.

Altogether, the grammar thus constructed requires at most $4b + (6j+8)t$ symbols. With $b = 4^j$ and $t \leq \log_b n$, we obtain

$$\begin{aligned} \mathtt{symb}(G_n) &\leq 4^{j+1} + 6j \log_{4^j} n + 8 \log_{4^j} n \\ &= 4^{j+1} + 6 \log_4 n + \frac{8}{j} \log_4 n. \end{aligned}$$

Setting $j = \frac{1}{2}\lfloor \log_4 \log_4 n \rfloor$, this evaluates to $6 \log_4 n + O(\frac{\log n}{\log \log n})$ as desired. □

Notice that in the above proof, we have $j > 1$ only for $n \geq 4^{4^4} = 2^{512}$, which is approximately $1.3 \times 10^{154}$.

Another interesting aspect is that the grammar constructed in the proof of [2, Thm. 11] has a number of $\log_3(n) + o(\log n)$ productions, and the sum of the lengths of the right-hand sides is $3 \cdot \log_3(n)$. This accounts for a symbol complexity of $5 \cdot \log_3(n) + o(\log n)$, which is off the optimal bound by a factor(!) of roughly $\frac{5 \cdot \log_3 n}{6 \cdot \log_4 n} \approx 1.05$. *Vice versa*, let us count the sum of the right-hand sides of the grammar we constructed in Theorem 13: the sum is $4 \cdot \log_4 n + o(\log n)$, whereas the optimal bound is approximately $3 \cdot \log_3 n$, as proved in [2, Thm. 11]. Regarding the sum of the right-hand sides, our new grammar is off the optimal bound by a factor of $\frac{4 \cdot \log_4 n}{3 \cdot \log_3 n} \approx 1.06$. This nicely illustrates how already a small variation in the definition of *size* of a context-free grammar can largely affect what kinds of grammars we consider optimal: will a strategy based on tripling the word length, or rather, one based on quadrupling, yield the smallest grammar?

We observe a similar discrepancy when restricting to grammars in Chomsky normal form, as done, e.g., in [16]. Let us again consider the sum of the right-hand sides. It is plausible that a smallest grammar $G$ in Chomsky normal form for $L_n$ has $m_G = 1 + 2 \cdot \log_2 n$ when $n$ is a power of two, and $\frac{2 \cdot \log_2 n}{3 \cdot \log_3 n} \approx 1.06$.

**Acknowledgment**

Thanks to Christian Rauch for his comments on an earlier draft of this paper. Moreover, the authors thank the anonymous referees for their helpful comments that improved the presentation and quality of this paper.

## References

1. Bar-Hillel, Y., Perles, M., Shamir, E.: On formal properties of simple phrase structure grammars. Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung **14**, 143–177 (1961)
2. Charikar, M., Lehman, E., Liu, D., Panigrahy, R., Prabhakaran, M., Sahai, A., Shelat, S.: The smallest grammar problem. IEEE Transactions on Information Theory **51**(7), 2554–2576 (July 2005). `https://doi.org/10.1109/TIT.2005.850116`
3. Dassow, J.: Descriptional complexity and operations—two non-classical cases. In: Pighizzini, G., Câmpeanu, C. (eds.) Proceedings of the 19th International Workshop on Descriptional Complexity of Formal Systems. pp. 33–44. No. 10316 in LNCS, Springer, Milano, Italy (July 2017). `https://doi.org/10.1007/978-3-319-60252-3_3`
4. Dassow, J., Harbich, R.: Production complexity of some operations on context-free languages. In: Kutrib, M., Moreira, N., Reis, R. (eds.) Proceedings of the 14th Workshop on Descriptional Complexity of Formal Systems. pp. 141–154. No. 7386 in LNCS, Springer, Braga, Portugal (July 2012). `https://doi.org/10.1007/978-3-642-31623-4_11`
5. Dassow, J., Stiebe, R.: Nonterminal complexity of some operations on context-free languages. Fundamenta Informaticae **83**(1–2), 35–49 (2008)
6. Ginsburg, S., Lynch, N.: Size complexity in context-free grammars forms. Journal of the ACM **23**(4), 582–598 (1976)
7. Gruber, H., Holzer, M., Rauch, C.: The pumping lemma for context-free languages is undecidable. In: Day, J.D., Manea, F. (eds.) Proceedings of the 28th International Conference on Developments in Language Theory. pp. 141–155. No. 14791 in LNCS, Springer, Göttingen, Germany (August 2024). `https://doi.org/10.1007/978-3-031-66159-4_11`
8. Gruska, J.: On the size of context-free grammars. Kybernetika **8**(3), 213–218 (1972)
9. Halmos, P.: Problems for Mathematicians, Young and Old. No. 12 in The Dolciani Mathematical Expositions, Mathematical Association of America (1991)
10. Harbich, R.: Regel- und Symbolkomplexität kontextfreier Sprachen unter ausgewählten Operationen. Ph.D. thesis, Otto-von-Guericke University Magdeburg, Germany (2019)
11. Harrison, M.A.: Introduction to Formal Language Theory. Addison-Wesley (1978)
12. Harrison, M.A., Yehudai, A.: Eliminating null rules in linear time. The Computer Journal **24**(2), 156–161 (1981). `https://doi.org/10.1093/comjnl/24.2.156`

13. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
14. Lange, M., Leiß, H.: To CNF or not to CNF? An efficient yet presentable version of the CYK algorithm. Informatica Didactica **8** (2009)
15. Mengel, S., Vinall-Smeeth, H.: A lower bound on unambiguous context free grammars via communication complexity. In: Proceedigns of the 44th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems. ACM (2025), accepted for publication
16. Mieno, T., Inenaga, S., Horiyama, T.: RePair grammars are the smallest grammars for Fibonacci words. In: Bannai, H., Holub, J. (eds.) Proceedings of the 33rd Annual Symposium on Combinatorial Pattern Matching. Leibniz International Proceedings in Informatics, vol. 223, pp. 26:1–26:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2022). `https://doi.org/10.4230/LIPICS.CPM.2022.26`
17. Miller, R.E., Muller, D.E.: A problem of maximum consistent subsets. Research Report RC-240, IBM Research Center, Yorktown Heights, New York, USA (March 1960)
18. Moon, J.W., Moser, L.: On cliques in graphs. Israel Journal of Mathematics **3**, 23–25 (1965)
19. Rowell, J.T.: Solution sequences for the keyboard problem and its generalizations. Journal of Integer Sequences **18**(10), 15.10.7 (2015)