The CFG Complexity of Singleton Sets

Lance Fortnow*
Illinois Institute of Technology
lfortnow@iit.edu

William Gasarch†
University of Maryland at College Park
gasarch@umd.edu

June 6, 2024

Abstract

Let G be a context-free grammar (CFG) in Chomsky normal form. We take the number of rules in G to be the size of G. We also assume all CFGs are in Chomsky normal form.

We consider the question of, given a string w of length n, what is the smallest CFG such that $L(G) = \{w\}$? We rediscover proofs of previously known results.

1. For all w, $|w| = n$, there is a CFG of size with $O(n/\log n)$ rules, such that $L(G) = \{w\}$.
2. There exists a string w, $|w| = n$, such that every CFG G with $L(G) = \{w\}$ is of size $\Omega(n/\log n)$. We give two proofs of: one nonconstructive, the other constructive.

1 Introduction

For a string w, how hard is it to generate w? For Turing machines, this relates to Kolmogorov complexity, where we know that most w require a program of length nearly $|w|$ to print w. However there is no computational process that will find an infinite set of such w.

In this paper, we look at similar questions but using context-free grammars (henceforth CFG's) as our computational device. We will need a standard form for CFG's.

Def 1.1 A CFG is in *Chomsky Normal Form* if every rule is in one of the following forms:

1. $A \to BC$ where A, B, C are all nonterminals.

*College of Computing, Ill. Inst. of Tech., IL 60616
†Dept. of Comp. Sci., Univ.of Maryland, MD 20742

2. $A \rightarrow \sigma$ where A is a nonterminal and $\sigma \in \Sigma$.

3. $S \rightarrow e$ where S is the start symbol and e is the empty string.

Henceforth all CFG's are assumed to be in Chomsky Normal Form.

It is well known that for every CFL L there exists a CFG G in Chomsky normal form that generates L. For a proof of this, and more background on formal language theory, see Sipser [6].

Notation 1.2 The *size* of a CFG is the number of rules.

In this paper we give proofs of these previously known results.

Theorem 1.3

1. *For all $w \in \{0,1\}^n$ we give a simple construction of a CFG for $\{w\}$ of size $O(n/\log n)$.*
2. *We show that for every n there exists a string w of length n such that every CFG for $\{w\}$ has size at least $\Omega(n/\log n)$ via a nonconstructive proof.*
3. *We show that if w is a de Bruijn sequence then $\{w\}$ requires a context-free grammar of size $\Omega(n/\log n)$.*

Along the way we post open questions about the sizes of CFG's for $\{w\}$.

1.1 Historical Note

These questions were studied in the past in terms of word chains. A word chain for w is a sequence of strings $w_1, \ldots, w_m = k$ where each w_i is either a letter of the input alphabet or the concatenation of w_j and w_k for $1 \leq j, k < i$. We leave it to the reader to see that the size of the minimum CFG for w is equivalent to the shortest word chain for w.

Lohrey has written a survey of these results [5]. We give specific citations for results as we discuss them later in the paper.

2 General Theorems about the Size of CFGs for $\{w\}$

The following theorem is easy to show.

Theorem 2.1

1. *Let $w = 0^n$. There is a CFG of size $O(\log n)$ for $\{w\}$.*
2. *Let $w \in \{0,1\}^n$. Any CFG for $\{w\}$ has size $\Omega(\log n)$.*

3. *Let $w \in \{0,1\}^n$. There is a CFG for $\{w\}$ of size $\leq n+1$.*

Part 1 of the following theorem has been proven independently by Berstel & Brelek [1].

Theorem 2.2

1. *Let $w \in \{0,1\}^n$. There is a CFG for $\{w\}$ of size $O(n/\log n)$.*
2. *Let $s_1, \ldots, s_k$ be a sequence of natural numbers. Assume k is even (for k odd a similar theorem holds). Let $w = 0^{s_1}1^{s_2}\cdots 0^{s_k}1^{s_k}$. There is a CFG of size $O(\sum_i^k \log s_i)$ for $\{w\}$.*
3. *Assume n is a square (if its not the statement still holds but is messier). Let $w = 0^1 1^2 0^3 \cdots 1^{\sqrt{n}}$. Note that $|w| = \Theta(n)$. There is a CFG of size $O(\sqrt{n}\log n)$ for $\{w\}$.*

Proof:
1) Let $k = \frac{\log n}{2}$. For simplicity we will assume that $k \in \mathbb{N}$ and $n/k \in \mathbb{N}$. If either is false then a similar proof applies.

We use Theorem 2.1 (3) throughout this proof.

1. For *every* string $x \in \{0,1\}^k$ form a CFG for $\{x\}$ of size $k+1$ with start symbol S_x. The total number of rules is $2^k(k+1) = k2^k + k$. We put *all* of these rules into our CFG. (We may not need all of these rules but it will not help our bound to toss out those we don't use.)
2. Let $w = x_1 \cdots x_{n/k}$ where, for all i, $|x_i| = k$.
3. Form a CFG G for $S_{x_1}\cdots S_{x_{n/k}}$. This will take $\frac{n}{k}+1$ rules. Put the rules for that CFG into our CFG. Note that G will generate {w}.

The total number of rules in out CFG is

$$k2^k + k + \frac{n}{k} + 1 = O\left(\sqrt{n}\log n + \frac{n}{\log n}\right) = O\left(\frac{n}{\log n}\right).$$

2) We use Theorem 2.1 (1) and (3) throughout this proof.

For $1 \leq i \leq k$, i even, let G_i be a CFG for 0^{s_i} of size $O(\log s_i)$. Let its start state be S_i. For $1 \leq i \leq k$, i odd, let H_i be a CFG for 1^{s_i} of size $O(\log s_i)$. Let its start state be T_i. Let G' be a CFG for $S_1T_1\cdots S_kT_k$ of size $k+1$ with start symbol S. Let G be the union of the CFG's G_i's, H_i's, and G'. Take S to be the start symbol. This grammar clearly generates the desired strings. The size of G is

$$k + 1 + O(\sum_{i=1}^{k} \log s_i) \leq O(\sum_{i=1}^{k} \log s_i).$$

3) By Part 2 there is a CFG for the desired language of size

$$O(\sum_{i=1}^{\sqrt{n}} \log i) \leq O(\sqrt{n} \log n).$$

∎

The following questions arise:

- Is there a string w such that any CFG for $\{w\}$ is large (for some definition of large). (Spoiler Alert: Yes.)
- Is there a *natural string* w such that any CFG for $\{w\}$ is large (for some definitions of natural and large). Theorem 2.2 (3) can be considered a failed attempt at getting a natural string w such that $\{w\}$ requires a large CFG. The string $w = 0^1 1^2 0^3 \cdots 1^{\sqrt{n}}$ is a candidate; however, it seems hard to prove that it requires a CFG of size $\Omega(\sqrt{n} \log n)$ or even any superlogarithmic lower bound.

Open Problem 2.3 Let $w = 0^1 1^2 0^3 \cdots 1^{\sqrt{n}}$. Obtain a lower bound $f(n)$ on the smallest CFG for $\{w\}$ such that is $\log n \ll f(n)$.

3 Strings w Such that any CFG for $\{w\}$ is of size $\Omega(n/\log n)$

Notation 3.1 Let $x, y \in \{0,1\}^*$.

1. $C(x)$ is the Kolmogorov complexity of x. (We assume some model of computation but note that if we chose a different one it would only affect $C(x)$ by an additive constant.)
2. $C(x \mid y)$ is the conditional Kolmogorov Complexity of x given y. (Same model comments apply here.)

For more background on Kolmogorov complexity see Li and Vitanyi [4].
The lower bound in the following theorem seems to be new.

Theorem 3.2 *There is a function $w : \mathbb{N} \to \{0,1\}^*$ such that, for all $n \in \mathbb{N}$, the following hold:*

1. *$|w(n)| = n$.*
2. *There is a CFG for $\{w(n)\}$ of size $O(n/\log n)$. This follows from Theorem 2.2 (1).*
3. *Any CFG that generates $\{w(n)\}$ has size $\Omega(\frac{n}{\log n})$.*

Proof: We define the function w as follows: $w(n)$ is the lexicographically least string of length n such that $C(w(n)) \geq n$. (The lex-least is not needed and is only there for definiteness.) We denote $w(n)$ by w.

Let G be a CFG that generates $\{w\}$. Let s be the size (number of rules) of G. If there are s rules, then each nonterminal can be represented with $O(\log s)$ bits. Hence each rule can be represented with $O(\log s)$ bits. Therefore the CFG can be represented with $O(s \log s)$ bits.

We use G to create a Turing Machine of size $O(s \log s)$ that, on input the empty string, outputs w:

Try all possible derivations to generate a string. The first time a string is generated, output it and stop.

Since $C(w) \geq n$ we have $n \geq \Omega(s \log s)$, hence $s \geq \Omega(\frac{n}{\log n})$. ∎

The function $w : \mathbb{N} \to \{0,1\}^*$ is not computable since one cannot compute strings of high Kolmogorov complexity. We can just search all strings and CFGs to find a w such that $\{w\}$ requires large CFGs but this is computationally expensive and not particularly natural.

In the next section we prove a version of Theorem 3.2 where the function w is efficiently computable and the strings produces are natural.

4 Efficiently Computable Strings w Such that any CFG for $\{w\}$ is of size $\Omega(n/\log n)$

In this section we show that de Bruijn sequences require large context-free grammars. We later comment on similar results that are known

Def 4.1 Let $k \in \mathbb{N}$. A *de Bruijn sequence of order k* is a binary sequence of length $n = 2^k$ where every string of length k appears exactly once as a substring, allowing wraparound.

Note 4.2 De Bruijn sequences are natural in that they have been rediscovered many times and have many applications. See the Wikipedia page on them for more information.

Example 4.3

1. 01 is a de Bruijn sequence of order 1.
2. 0011 is a de Bruijn sequence of order 2. Note that 10 is wraparound.
3. 10111000 is a de Bruijn sequence of order 3.
4. 0000111101100101 is a de Bruijn sequence of order 4.

5. It is known that, for every k, there is a de Bruijn sequence of length 2^k.

Lemma 4.4 (Elder et. al [3]) *There is an algorithm that will, on input 1^n, where n is a power of 2, output a de Bruijn sequence of length n, in time $O(n)$.*

The following theorem was obtained by Domaratzki et al. [2] independently.

Theorem 4.5

1. *Let $k \in \mathbb{N}$. Let w be a de Bruijn sequence of length n and order $k = \log n$. Any CFG for $\{w\}$ is of size $\Omega(n/\log n)$.*
2. *There is a function $w : \mathbb{N} \to \{0,1\}^*$ such that, for all $n \in \mathbb{N}$, the following hold:*
   (a) *$|w(n)| = n$.*
   (b) *There is a CFG for $\{w(n)\}$ of size $O(n/\log n)$. This follows from Theorem 2.2 (1).*
   (c) *Any CFG that generates $\{w(n)\}$ has size $\Omega(\frac{n}{\log n})$.*
   (d) *There is a linear time algorithm for w.*

Proof: Part 2 follows from Part 1 and Lemma 4.4. Hence we just proof Part 1.

Let G be a CFG for $\{w\}$. Look at the parse tree for the derivation of w. We assume that every nonterminal appears in the parse tree. No terminal A can have the same terminal A below it in the derivation tree or G would generate an infinite language.

Let A be a nonterminal that appears at least twice in the parse tree. Let z be such that $A \Rightarrow z$. We show that $|z| \leq k-1$. Assume $|z| \geq k$. Then the first k bits of z appear twice in w. This contradicts w being a de Bruijn sequence of order k. We need the contrapositive: If $B \Rightarrow z$ and $|z| \geq k$ then B appears once in the parse tree.

Let $A_1, \ldots, A_L$ be the multiset of nonterminals in the parse tree such that (1) A_i generates a string of length $\leq k-1$, and (2) the parent of A_i generates a string of length $\geq k$. Let $B_1, \ldots, B_M$ be the parents of $A_1, \ldots, A_L$. Note that $L \geq \frac{n}{k} = \frac{n}{\log n}$; however, the number of distinct nonterminals in $\{A_1, \ldots, A_L\}$ might be quite small. But note that $M \geq L/2$ and the nonterminals in $\{B_1, \ldots, B_M\}$ are all distinct. Hence there are $\geq \frac{n}{2} \log n$ nonterminals. Since each nonterminals is the LHS of some rule, there are at $\geq \frac{n}{2} \log n$ rules. ∎

Berstel & Brelek [1] showed the following, in our language:

Theorem 4.6

1. *Let $k \in \mathbb{N}$. Let $n = 2^k$. Let $\Sigma| = 1$ and let $\$ \in \Sigma$. Let w be a the string*

$$\$u_1\$u_2 \cdots \$u_{(q-1)^k}.$$

Any CFG for $\{w\}$ is of size $\Omega(n/\log n)$.

2. *Let $q \geq 3$. There is a function $w : \mathbb{N} \to \{0, 1, \ldots, q\}^*$ such that, for all $n \in \mathbb{N}$, the following hold:*

   (a) $|w(n)| = n$.

   (b) There is a CFG for $\{w(n)\}$ of size $O(n/\log n)$. This follows from Theorem 2.2 (1).

   (c) Any CFG that generates $\{w(n)\}$ has size $\Omega(\frac{n}{\log n})$.

   (d) There is a linear time algorithm for w.

Theorem 4.5 and 4.6 are incomparable. Theorem 4.5 uses an alphabet of size 2, but the complexity of the function is quasilinear. In Theorem 4.6 the function has linear complexity, but the alphabet has to be size ≥ 3.

5 Intermediary Complexity

We now have two extremes:

- If $w = 0^n$ then $\{w\}$ can be generated by a CFG of size $O(\log n)$.
- If w is Kolmogorov random or a de Bruijn sequence then any CFG that generates $\{w\}$ is of size $\Omega(\frac{n}{\log n})$.

Are there w such that $\{w\}$ can be generated by a CFG of size intermediary between $O(\log n)$ and $O(n)$? Yes. We won't dwell on this, but the key is to take a string of the form $w0^{n-f(n)}$ where (1) f is chosen carefully depending on which intermediary function you want, and (1) w is a Kolmogorov random string or de Bruijn sequence of length $f(n)$.

6 Acknowledgements

We would like to thank Cheng-Yuan Lee for proofreading. We would like to thank Markus Lohrey and Giovanni Pighizzini for giving us valuable pointers to the literature.

References

[1] J. Berstel and S. Brelek. On the length of word chains. *Information Processing Letters*, 26:23–28, 1987.

[2] M. Domaratzki, G. Pighizzini, and J. O. Shallit. Simulating finite automata with context-free grammars. *Inf. Process. Lett.*, 84(6):339–344, 2002.

[3] C. Eldert, H. M. Gurk, H. J. Gray, and M. Rubinoff. Shifting counters. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, 77(1):70–74, 1987.

[4] Li and Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, New York, Heidelberg, Berlin, 2008. This is the fourth edition.

[5] M. Lohrey. Algorithmics on slp-compressed strings: A survey. *Groups Complex. Cryptol.*, 4(2):241–299, 2012. `https://www.eti.uni-siegen.de/ti/veroeffentlichungen/12-survey.pdf`.

[6] M. Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 2012.