On the Sizes of DFAs, NDFAs, CFGs, CSLs, and TMs: A Survey

by William Gasarch

1 Introduction

Let Σ be a finite alphabet. All of our languages will be a subset of Σ^*. Recall the following types of languages.

1. A language L is *regular* (henceforth REG) if it satisfies any of the following three equivalent conditions. (1) L can be recognized by a deterministic finite automata (henceforth DFA). We measure the size of a DFA by the number of states. (2) L can be recognized by a nondeterministic finite automata (henceforth NDFA). We measure the size of an NDFA by the number of states. (3) There is a regular expression (henceforth R.E.) that generates L. We measure the size of a R.E. by its length.

2. A language L is *deterministic context free* (henceforth DCFL) if it can be recognized by a deterministic push down automata (henceforth DPDA). We measure the size of a DPDA by the sum of the number of states and the number of symbols in its stack alphabet.

3. A language L is *context free* (henceforth CFL) if it satisfies any of the following two equivalent conditions. (1) L can be recognized by a pushdown automata (henceoforth PDA). We measure the size of a PDA by the sum of the number of states and the number of symbols in its stack alphabet. (2) There is a context free grammar (henceforth CFG) that generates L. We will assume that the CFG is in Chomsky Normal Form (all productions of the form either $A \rightarrow BC$ or $A \rightarrow \sigma$ where A, B, C are nonterminals and $\sigma \in \Sigma$.) We measure the size of a CFG by the number of nonterminals in it. This is poly-related to any other reasonable measure.

4. A language L is *context sensitive* (henceforth CSL) if it satisfies any of the following two equivalent conditions. (1) L can be recognized by a nondeterministic Turing machine that

uses $O(n)$ space. (2) There is a context sensitive grammar (henceforth CSG) that generates L. (Rules are of the form $\alpha \rightarrow \beta$ such that α has at least one nonterminal, and $|\alpha| \leq |\beta|$. We will assume that the CSG is in Chomsky Normal Form (all productions are of the form $AB \rightarrow CD$, $A \rightarrow BC$, $A \rightarrow \sigma$). We measure the size of a CSG by the number of nonterminals in it. This is poly-related to any other reasonable measure.

5. A language L is *Decidable* (henceforth DEC) if there is a Turing machine (henceforth TM) M such that (1) if $x \in L$ then $M(x) = 1$, and (2) if $x \notin L$ then $M(x) = 0$. Throughout this paper we assume we have a standard list of Turing machines $M_1, M_2, \ldots$. We measure the size of TM M_e by e.

It is well known that

$$\text{REG} \subset \text{DCFL} \subset \text{CFL} \subset \text{CSL} \subset \text{DEC}.$$

Our concern is with the *size* of the DFA, DPDA, CFG, CSG, TM. For example, let L be regular. Is it possible that there is a CFG for L that is much smaller than the minimal DFA for L? For all adjacent pairs above we will consider these questions.

Recall that DFA's, NDFA,'s and R.E.'s have the exact same computing power. It is known that for every NDFA of n states there is an equivalent DFA with at most 2^n states. Is this tight? What about going from DFA's to R.E.'s? We consider questions of this type.

Recall that REG is closed under many operations (union, intersection, complementation, concatenation, Kleene star, reversal). For some of these constructions the DFA increases in size exponentially. Is this necessary? We consider questions of this type.

Def 1.1 Let $\mathcal{M}$ and $\mathcal{M}'$ be two classes of devices such that every language in $\mathcal{M}$ is also in $\mathcal{M}'$. (e.g., DFA's and DPDA's). A *bounding function for* $(\mathcal{M}, \mathcal{M}')$ is a function f such that for all $L \in \mathcal{M}$, if L is represented by a device of size n in $\mathcal{M}'$ then it is represented by a device of size

$\leq f(n)$ in $\mathcal{M}$.

Examples The following bounding functions can be found in all formal language theory textbooks unless otherwise noted.

1. $f(n) = 2^n$ is a bounding function for (DFA,NDFA). This is called *the powerset construction*.

2. $f(n) = O(n^4)$ is a bounding function for (CFG,PDA).

3. $f(n) = n + O(1)$ is a bounding function for (PDA,CFG).

4. $f(n) = n^{n^{n^n}}$ is a bounding function for (REG,DPDA). Stearns [10] proved this. This result is not in any textbooks that we have looked at.

BILL- INSERT WHAT IS KNOWN ABOUT OTHER CONVERSIONS: CSG AND LBA

Notation 1.2

1. If w is a string and $\sigma \in \Sigma$ then $\#_\sigma(w)$ is the number of σ's in w

2. If $n \in \mathsf{N}$ then $[n]$ is the set $\{1, \ldots, n\}$.

2 Summary

3 Useful Lemmas

Lemma 3.1 *Let X, Y, Z be nonterminals. Let Σ be a finite alphabet.*

1. *For all $n \geq 2$ there is a CFG of size $O(\log n)$ that generates $\{Y^n\}$.*

2. *For all $n \geq 2$ there is a CFG of size $O(\log n)$ that generates $\{XY^n\}$.*

3. *For all $n \geq 2$ there is a CFG of size $O(\log n)$ that generates $\{Y^nZ\}$.*

4. *For all $n \geq 2$ there is a CFG of size $O(\log n)$ that generates $\{XY^nZ\}$.*

5. *For all $n \geq 2$ there is a CFG of size $O(\log n)$ that generates $\{a,b\}^n$.*

6. *For all $n \geq 2$ there is a CFG of size $O(\log n)$ that generates $\{a,b\}^{\leq n}$.*

Proof:

1) We show that there is a CFG of size $\leq 2\lg n$ that generates $\{Y^n\}$ by induction on n.

If $n = 2$ then the CFG for $\{YY\}$ is $S \rightarrow YY$ which has $2 = 2\lg 2$ nonterminals.

If $n = 3$ then the CFG for $\{YYY\}$ is $\{S \rightarrow Y_1Y, Y_1 \rightarrow YY$ which has $3 \leq 2\lg 3$ nonterminals.

Assume that for all $m < n$ there is a CFG of size $\leq 2\lg m$ for $\{Y^m\}$. We prove this for n.

1. n is even. Let G' be the CFG for $\{Y^{n/2}\}$ with the start symbol replaced by S'. The CFG G for $\{Y^n\}$ is the union of G' and the one rule $S \rightarrow S'S'$. This CFG has one more nonterminal than G'. Hence the number of nonterminals in G is

$$\leq 2\lg(n/2) + 1 = 2(\lg n - 1) + 1 = 2\lg n - 1 \leq 2\lg n.$$

2. n is odd. Let G' be the CFG for $\{Y^{(n-1)/2}\}$ with the start symbol replaced by S'. The CFG G for $\{Y^n\}$ is the union of G' and the two rules $S \rightarrow YS''$ and $S'' \rightarrow S'S'$. This CFG has two more nonterminal than G'. Hence the number of nonterminals in G is

$$\leq 2\lg((n-1)/2) + 2 = 2(\lg(n-1) - 1) + 2 = 2\lg(n-1) - 2 + 2 = 2\lg(n-1) \leq 2\lg n.$$

2,3,4,5,6) These are easy modifications of the CFG in part 1.

▮

4 Languages with Small NDFA that Require a Large DFA ($|\Sigma| \geq 2$)

The following is well known. It is a corollary of the Myhill-Nerode Theorem which is in most formal language theory textbooks.

Lemma 4.1 *Let L be a regular language via DFA $M = (Q, \Sigma, \delta, s, F)$. Assume $x, y, z \in \Sigma^*$ such that $xz \in L$ but $yz \notin L$. Then $\delta(s, x) \neq \delta(s, y)$.*

According to Meyer and Fischer [7] the following example is due to Patterson.

Theorem 4.2 *For all n there exists a language L_n such that*

1. *Any DFA for L_n requires 2^n states.*
2. *There is an NDFA for L_n that has $n + 2$ states.*

Proof: Let $\Sigma = \{a, b\}$. Let

$$L_n = \{a, b\}^* a \{a, b\}^n.$$

A string w is in L_n iff there is an a in w that is exactly n letters from the end.

1) Let M be a DFA for L_n. We show that M has at least 2^n states. Map every string x of length n to $\delta(s, x)$. This map is injective: Let $x, y \in \Sigma^n$. Since $x \neq y$ let i be the first place they differ. We can assume $x = uax'$ and $y = uby'$ where $|u| = i - 1$ and $|x'| = |y'| = n - i$. Let $z = a^i$. Since $xz \in L_n$ and $yz \notin L_n$, by Lemma 4.1 $\delta(s, x) \neq \delta(s, y)$.

2) We describe an NDFA that accepts L_n: The start state has a self-loop on a or b. Also, on an a the NDFA goes to a new state. So that state means *I just read an a*. There are n more states that indicate *I just read i chars after the a*. The last one is final. This NDFA has $n + 2$ states.

BILL- HAVE KARTHIK ADD A PICTURE OF THE NDFA HERE.

∎

Theorem 4.2 gives an NDFA of size $n + 2$ and a DFA of size 2^n. We noted earlier that a bounding function for (NDFA,DFA) is $f(n) = 2^n$. The question arises: is there a language whose NDFA is of size n and whose DFA is of size 2^n. Yes!

The following theorem is due to Meyer and Fischer [7].

Theorem 4.3 *For all n there exists a language L_n such that*

1. *Any DFA for L_n requires 2^n states.*
2. *There is an NDFA for L_n that has n states.*

Proof: Let $\Sigma = \{a, b\}$. We describe an NDFA for L_n: There are n states labeled $\{0, \ldots, n-1\}$. If you are in state i and a b comes in then goto state $i+1$ (mod n). If you are in state $1 \le i \le n-1$ and an a comes in then goto both state i and state 0. (If you are in state 0 and an a comes in there is no instruction.) The start state and the final state are both 0.

BILL AND KARTHIK. DRAW PICTURE AND PROVE THAT IT WORKS. KARTHIK- IF YOUR EXAMPLE IS EASIER TO WORK WITH THEN WE"LL DO THAT ONE EITHER INSTEAD OR IN ADDITION TO THIS ONE.

▮

5 Languages with Small NDFA that Require a Large DFA ($|\Sigma| = 1$)

Is there a regular language over a unary alphabet whose NDFA is much smaller than its DFA? Yes! The result in this section was communicated to the author by Richard Beigel. This is likely its first time in print.

BILL- ASK RICHARD IF THIS IS TRUE.

Theorem 5.1 *For all n there exists a unary regular language L_n such that*

1. *Any DFA for L_n requires $e^{(1+o(1))n \log n}$ states.*

2. *There is an NDFA for L_n that has $O\big(\frac{n^2}{\log n}\big)$ states.*

Proof: Let $p_1, p_2, \ldots, p_L$ be all of the primes that are $\le n$. Let

$$L_n = \{a^m : m \not\equiv 0 \pmod{\prod_{i=1}^{L} p_i}\} = \{a^m : (\exists i \le L)[m \not\equiv 0 \pmod{p_i}]\}.$$

1) Let M be a DFA for L_n. Let $Z = \prod_{i=1}^{L} p_i$. We map $\{0, \ldots, Z-1\}$ to states of M. Map a^i to $\delta(s, a^i)$.

Let $0 \le i < j \le Z-1$. Since $a^i a^{Z-j} = a^{Z+i-j} \notin L_n$ and $a^j a^j a^{Z-j} = a^Z \in L_n$, we know that, by Lemma 4.1, $\delta(s, a^i) \ne \delta(s, a^j)$. Hence the map is injective. Therefore there are at least $Z = e^{(1+o(1))n\log n}$ states.

2) The following NDFA accepts L_n with $\sum_{i=1}^{L} p_i = O\big(\frac{n^2}{\log n}\big)$ states.

For $1 \le i \le L$ let D_i be the DFA that accepts $\{a^m : m \not\equiv 0 \pmod{p_i}\}$. Let s_i be the start state of D_i. Note that this has p_i states. Let M be the NDFA that has, from the start state, L different e transitions, each one to a different s_i. M clearly accepts L_n.

BILL- TELL KARTHIK WE NEED A PICTURE OF THE NDFA

▮

Open Problem 5.2

1. Does there exist a regular unary language that has a small NDFA, requires a large DFA, but does not require the prime number theorem to prove these bounds.

2. Does there exist a regular unary language where the gap between the NDFA and DFA is larger than that in Theorem 5.1?

3. Does there exist a regular unary language where the NDFA is of size n and the DFA requires size 2^n?

6 Languages with Small DFAs that Require a Large Reg Expression

The results in this chapter are due to Ehrenfeucht and Zeiger [1].

BILL- READ THIS PAPER AND PUT STUFF HERE

7 Small R.E implies Small NDFA

BILL- PUT THE R.E. TO NDFA CONSTRUCTION HERE

8 Pairs of Languages with Small DFAs such that L_1L_2 Requires a Large DFA

BILL- THIS DOESN"T QUITE LOOK RIGHT- SEEMS TO VIOLTE THE UPPER BOUND. I"LL LOOK AT IT LATER. The usual proof that if L_1 and L_2 are regular than L_1L_2 is regular can be demonstrated by this picture

BILL- HAVE KARTHIK PUT A PICTURE HERE.

IF L_1's DFA has n_1 states and L_2's DFA has n_2 states then we obtain an NDFA for L_1L_2 with $n_1 + n_2$ states. By the powerset construction we then obtain a DFA with at most $2^{n_1+n_2}$ states. However, S. Yu et al. [11] have analyzed this construction more carefully and obtained the following.

Theorem 8.1 *If L_1 is regular via a DFA with n_1 states and L_2 is regular via a DFA with n_2 states then L_1L_2 is regular via a DFA with $n_1 2^{n_2} - 2^{n_2=1}$ states.*

Proof:

BILL- PUT IN PROOF LATER

▮

We present two lower bounds. The first one is not optimal but is simple and good for teaching. The second one, due to S. Yu et al. [11], is optimal.

Theorem 8.2 *For all n there exists languages $L_{n,1}$ and $L_{n,2}$ such that*

1. *There is a DFA for $L_{n,1}$ with 2 states.*
2. *There is a DFA for $L_{n,2}$ with n states.*
3. *Any DFA for $L_{n,1}L_{n,2}$ requires 2^n states.*

BILL- THIS DOESN"T QUITE LOOK RIGHT- SEEMS TO VIOLTE THE UPPER BOUND. I"LL LOOK AT IT LATER.

Proof: Let

$L_{n,1} = \{a,b\}^*$. The min DFA for $L_{n,1}$ has 2 states. ($L_{n,1}$ does not depend on n.)

$L_{n,2} = a\{a,b\}^n$. The min DFA for $L_{n,2}$ has n states.

Then

$L_{1,n}L_{2,n} = \{a,b\}^*a\{a,b\}^n$.

By the proof of Theorem 4.2 any DFA for $L_{n,1}L_{n,2}$ requires 2^n states. ∎

Theorem 8.3 *For all n_1, n_2 there exists languages L_{n_1} and L_{n_2} such that*

1. *There is a DFA for L_{n_1} with n_1 states.*
2. *There is a DFA for L_{n_2} with n_2 states.*
3. *Any DFA for $L_{n_1}L_{n_2}$ requires $n_1 2^{n_2} - 2^{n_2-1}$ states.*

Proof:

BILL- PUT IN THIS PROOF

KARTHIK- THIS NEEDS A PICTURE FROM THE PAPER I REFERENCE ABOVE.

∎

9 Languages with Small DFA such that their Kleene Closure Requires a Large DFA

The usual proof that if L are regular than L^* is regular can be demonstrated by this picture

BILL- HAVE KARTHIK PUT A PICTURE HERE.

IF L's DFA has n states then we obtain an NDFA for L^* with $n+1$ states. By the powerset construction we then obtain a DFA with at most 2^{n+1} states. However, S. Yu et al. [11] have analyzed this construction more carefully and obtained the following.

Theorem 9.1 *Let L be a regular language via a DFA with n states and k final states that are not start states. Then L^* has a DFA with $\le 2^{n-1} + 2^{n-k-1}$ states.*

Proof:

Let L be a regular language recognized by DFA $M = (Q, \Sigma, \delta, s, F)$. Assume $|Q| = n$. We will assume that $s \notin F$. The reader can work out the case where $s \in F$.

The NDFA (pictured above) for L^* is $M' = (Q \cup \{s'\}, \Sigma, \Delta, s', F \cup \{s'\})$ where Δ is defined as follows

$\Delta(s', e) = s$

$\Delta(f, e) = s$ for all $f \in F$.

$\Delta(q, \sigma) = \delta(q, \sigma)$ for all $q \in Q$, $\sigma \in \Sigma$.

Let N be the NDFA obtained by the powerset construction. We show that after getting rid of unreachable states, the number of states in N is $\le 2^{n-1} + 2^{n-k-1}$.

Recall that the states of N are powersets of $Q \cup \{s'\}$. Let $P \subseteq Q \cup \{s'\}$. We are wondering which P will be reachable in N. Note the following.

- If $(\exists f \in F \cap P)$ then $s \in P$ because $\Delta(f, e) = s$.
- The empty set is not reachable. Formally you can prove this by induction on the formation of reachable states.

- If P is reachable and $|P| \geq 2$ then $s' \notin P$. Formally you can prove this by induction on the formation of reachable states.

Using these facts the only states P that are potentially reachable are as follows.

1. $\{s'\}$. There is 1 of these.
2. Non empty subsets of Q that do not contain s or any element of F. There are $2^{n-k-1} - 1$ of these.
3. Non empty subsets of Q that contain s but no element of F. There are 2^{n-k-1} of these.
4. Non empty subsets of Q that contain at least one element of F (and hence contain s). There are $(2^k - 1)2^{n-k-1} = 2^{n-1} - 2^{n-k-1}$ of these.

The total number of states that are reachable is bounded by

$$1 + 2^{n-k-1} - 1 + 2^{n-k-1} + 2^{n-1} - 2^{n-k-1} = 2^{n-1} + 2^{n-k-1}.$$

∎

S. Yu et al. [11] showed that this is optimal in the $k = 1$ case.

Theorem 9.2 *For all n there exists language L_n such that*

1. *There is a DFA for L_n with n states, one of which is final. The final state is not the start state.*
2. *Any DFA for L_n^* requires $2^{n-1} + 2^{n-2}$ states.*

Proof:

BILL- ASK KARTHIK TO DRAW THIS DFA.

BILL-IS THERE AN EASIER EXAMPLE.

BILL- SUPPLY PROOF HERE

∎

10 Languages with Small DFAs such that their Reversal Requires a Large DFA

Def 10.1 If w is a string then w^R is that string written backwards (e.g., $(aaab)^R = baaa$). If L is a language then

$$L^R = \{w : w^R \in L\}.$$

The following theorem is well known.

Theorem 10.2 *If L is regular via a DFA with n states then L^R is regular via a DFA with 2^n states.*

Proof: L is regular via $M = (Q, \Sigma, \delta, s, F)$. We construct an NDFA for L^R.

Reverse all of the δ's in M. Formally

$M^R = (Q \cup \{s'\}, \Sigma, \Delta, s', s)$ where

$$\Delta(q, \sigma) = \{p : \delta(p, \sigma) = q\}.$$

$$\Delta(s', e) = F.$$

This yields an NDFA on n states. By the powerset construction there is a DFA with $\leq 2^n$ states.

▮

We show that this blowup is close to optimal.

BILL- CAN WE GET A BETTER EXAMPLE WHERE IS REALLY 2^n?

Theorem 10.3 *For all n there exists language L_n such that*

1. *There is a DFA for L with $n + 1$ states.*

2. *Any DFA for L^R requires 2^n states.*

Proof: Let

$$L_n = \{a,b\}^n a\{a,b\}^*.$$

$$L_n^R = \{a,b\}^* a\{a,b\}^n.$$

It is easy to see that L_n has a DFA on $n+1$ states. By the proof of Theorem 4.2 any DFA for L_n^R has size at least 2^n. ▮

BILL AND KARTHIK- CAN WE GE THIS TO BE OPTIMAL? WOULD NEED A DIFF L_n.

11 Languages with Small NDFAs such that their Compliment Requires a Large DFA ($|\Sigma| \geq 2$)

BILL AND KARTHIK- CAN WE GET A BIGGER SEP THEN IF HAVE $|\Sigma| \geq 2$.

12 Languages with Small NDFAs such that their Compliment Requires a Large DFA ($|\Sigma| \geq 1$)

Theorem 12.1 *For all n there exists a regular language L_n such that*

1. *Any NDFA for $\overline{L_n}$ requires $e^{(1+o(1))n\log n}$ states.*
2. *There is an NDFA for L_n that has $O(\frac{n^2}{\log n})$ states.*

Proof: Let $p_1, p_2, \ldots, p_L$ be all of the primes that are $\leq n$. Let $P = \prod_{i=1}^{L} p_i$. By the prime number theorem $P \sim e^{(1+o(1))n\log n}$.

$$L_n = \{a^m : m \equiv 0 \pmod{P}\} = \{a^m : (\forall i \leq L)[m \equiv 0 \pmod{p_i}\}.$$

1) Let M be an NDFA for L_n. Let $0 \leq m \leq P$. Map a^m to the least state $q \in \Delta(s, a^{m_1})$ such that $F \cap \delta(q, a^{P-m}) \neq \emptyset$. (So there is a path from q to a final state.)

We show that if $0 \leq m_1 < m_2 \leq P$ then a^{m_1} and a^{m_2} map to different states. Assume, by way of contradiction, that a^{m_1} and a^{m_2} both map to q. Then we have $q \in \delta(s, a^{m_1})$ and $F \cap \delta(q, a^{P-m_2}) \neq \emptyset$. Hence $\delta(s, a^{P+m_1-m_2}) \cap F \neq \emptyset$. Therefore $a^{P+m_1-m_2} \in L_n$ which is a contradiction.

Since the mapping is 1-1 there must be at least P states.

2)

$$\overline{L_n} = \{a^m : m \not\equiv 0 \pmod{P}\} = \{a^m : \exists i \leq L)[m \not\equiv 0 \pmod{p_i})\}.$$

We proved in Theorem 5.1 that this has an NDFA of size $\sum_{i=1}^{L} p_i = O(\frac{n^2}{\log n})$ ▮

Open Problem 12.2

1. Does there exist a regular unary language that has a small NDFA whose compliment requires a large NDFA, but does not require the prime number theorem to prove these bounds.

2. Does there exist a regular unary language where the gap between the NDFA for it and for its compliment is larger than that in Theorem 5.1?

3. Does there exist a regular unary language where the NDFA is of size n and the NDFA for its compliment requires size 2^n?

13 Languages that have a Small DPDA and require a Large NDFA

Stearns [10] showed that if L is a regular language which has a DPDA of n states and s stack alphabet then L has a DFA of size at most $s^{n^{n^n}}$ states. He proved a bit more than that. We state his entire theorem for future use.

Theorem 13.1 *There is an algorithm that will do the following. Given a DPDA P with n states and s stack alphabet:*

1. *Determine if $L(P)$ is regular.*
2. *If $L(P)$ is regular then output a DFA for $L(P)$ of size $\le s^{n^{n^n}}$*

Corollary 13.2 *(We separate out the parts of the size of a DPDA: n for the number of states and s for the stack alphabet size.) The function $f(s,n) = s^{n^{n^n}}$ is a bounding function for (DFA,DPDA).*

Theorem 13.3 *For all n, there exists a unary language L_n such that*

1. *Any NDFA for L_n requires 2^n states.*
2. *There is a DPDA for L_n with n states.*

Proof: Let L_n be the singleton set

$$L_n = \{a^{2^n}\}.$$

1) The proof that any NDFA for L_n is of size $\ge 2^n$ is similar to the proof in Theorem 12.1 that the language in that theorem required P states.

2) We give a DPDA for L_n. The stack alphabet is $A_1, \ldots, A_n$. There is only one state, hence we do not mention it.

On any input do the following

1. Put $A_1, A_2, \ldots, A_n$ on the stack while reading no input.
2. If there is an A_n on the top of the stack and the next input is a then read the input and pop the A_n.
3. If there is an A_i on top of the stack, $1 \le i \le n-1$, then replace it with $A_{i+1}A_{i+1}$.

Over time there will be exactly 2^n A_n's in the stack. The only way to accept is if there are exactly 2^n a's in the input. ▮

14 Languages that have a Small DPDA and Require a Very Large DFA

BILL- PUT IN FILE dpda.tex HERE

15 Languages that have a Small NDFA and Require a Large DPDA. Really!

DPDA's are more powerful than NDFA's since they have a stack. But NDFA's have nondeterminism. This can be used to obtain a case where the NDFA is small and the DPDA is provably large. Really! We will need to use a theorem from later in the paper.

We need a lemma. The proof is *not* in that many textbooks; however, it is in the text of Lewis and Papadimitriou [8].

Lemma 15.1 *If L is recognized by a DPDA of size s then $\overline{L}$ is recognized by a DPDA of size $s+1$.*

Theorem 15.2 *For all n there exists a language L_n such that*

1. *Any DPDA for L_n requires $\Omega\left(\frac{1.17^n}{n^{3/8}}\right)$.*
2. *There is an NDFA that recognizes L_n of size $O(n)$.*

Proof: Let $[n] = \{1, \ldots, n\}$. Let $PERM(n)$ be the set of all permutations of $[n]$. We view this as a language over the alphabet $[n]$. Note that it has $n!$ elements.

Let $L_n = \overline{PERM(n)}$

1) Let D be a DPDA for L_n of size s. Then, by Lemma 15.1 there is a DPDA for $\overline{L_n} = PERM(n)$ of size $s+1$. By the bounding functions given in the introduction there is a CFG for $PERM(n)$ of size $(s+1)^4 = O(s^4)$. By Theorem 20.7 any CFG for L_n has size $\Omega\left(\frac{1.89^n}{n^{3/2}}\right)$. Hence

$$s \geq \Omega\left(\left(\frac{1.89^n}{n^{3/2}}\right)^{1/4}\right) = \Omega\left(\frac{1.17^n}{n^{3/8}}\right)$$

2) Let $A_i = ([n] - \{i\})^*$ and $B_i = [n]^* i [n]^* i [n]^*$. A_i has an NDFA of size 1 (just one state with self loops for all alphabet symbols except i). B_i has an NDFA of size 3 (exercise). The NDFA that has an e-transition to all of the A_i and B_i has size $O(n)$. This NDFA recognizes L_n.

BILL- IF TIGHEN UP CFG TO PDA CAN IMPROVE ∎

16 Languages that have a Small CFG and Require a Large DPDA

We show an exponential size difference between CFG's and DPDA's. We will use a theorem proven later in this paper.

Theorem 16.1 *For all n there exists a language L_n such that*

1. *Any DPDA for L_n requires $\Omega(\frac{1.17^n}{n^{3/8}})$.*
2. *There is a CFG that recognizes L_n of size $O(n)$.*

Proof:
1) In Theorem 15.2 we showed that any DPDA for $PERM([n])$ requires size $\Omega(\frac{1.17^n}{n^{3/8}})$. Let $L_n = \overline{PERM([n])}$. By Lemma 15.1 L_n also requires size $\Omega(\frac{1.17^n}{n^{3/8}})$.

2) We could invoke a theorem about how a small NDFA leads to a small CFG. We choose instead to write down the CFG directly.

Let $A_i = ([n] - \{i\})^*$ and $B_i = [n]^* i [n]^* i [n]^*$.

A_i has a CFG of size $O(n)$

$$\begin{aligned} S &\implies [j]S \text{ for all } j \neq i \\ [j] &\implies j \text{ for all } j \neq i \end{aligned}$$

B_i has a CFG of size $O(1)$

$$
\begin{aligned}
S &\implies T[i]T[i]T \\
T &\implies j \text{ for all } j \in [n]
\end{aligned}
$$

∎

17 Languages that have a Small CFG and Require a Very Large DPDA

We show a double-exponential size difference between CFG's and DPDA's. We will use a theorem from later in this paper.

Theorem 17.1 *For all n there exists a language L_n such that*

1. *Any DPDA for L_n requires $\Omega\big(\frac{1.09^n}{n^{1/4}}\big)$.*
2. *There is a CFG that recognizes L_n of size $O(\log n)$.*

Proof: Let $W_n = \{ww : |w| = n\}$. Let $L_n = \overline{W_n}$

1) Let D be a DPDA for L_n of size s. Then, by Lemma 15.1 there is a DPDA for $\overline{L_n} = W_n$ of size $s+1$. By the bounding functions given in the introduction there is a CFG for W_n of size $(s+1)^4 = O(s^4)$ By Theorem 20.11 any CFG for W_n has size $\Omega\big(\frac{2^{n/2}}{n}\big)$.

Hence

$$
s \geq \Omega\bigg((\frac{2^{n/2}}{n})^{1/4} \bigg) = \Omega\bigg(\frac{2^{n/8}}{n^{1/4}}\bigg) = \Omega\bigg(\frac{1.09051^n}{n^{1/4}}\bigg) = \Omega\bigg(\frac{1.09^n}{n^{1/4}}\bigg)
$$

2) We present a CFG for L_n. Note that if $x \in L_n$ then either $|x| \leq 2n-1$ or there are two letters in x that are different and are exactly $n-1$ apart.

The CFG is the union of two CFG's. The first one generates all strings of length $\leq 2n-1$. By Lemma 3.1 there is such a CFG of size $O(\log n)$.

The second one generates all strings of length $\geq 2n$ where there are two letters that are different and exactly $n-1$ apart.

By Lemma 3.1 there is a CFG G' of size $O(\log n)$ generates all strings of length $n-1$. Let S' be its start symbol. G' will be part of our CFG G, though S' will not be the start symbol.

$S \rightarrow UaS'bU$

$S \rightarrow UbS'aU$

$U \rightarrow aU$

$U \rightarrow bU$

$U \rightarrow e$

Add all of the rules in G'.

This CFG clearly generates what we want and is of size $O(\log n)$. ▮

Corollary 17.2 *There exists a constant $\alpha > 1$ such that the following holds. For all n there exists a language L_n such that*

1. *Any DPDA for L_n requires $2^{2^{\Omega(\sqrt{n})}}$.*
2. *There is a CFG that recognizes L_n of size $O(n)$.*

18 Languages that have a Small CFG and Require a Ginormous DPDA

In Corollary 17.2 we obtained a language with CFG of size $O(n)$ but any DPDA requires size $2^{2^{\Omega(\sqrt{n})}}$. Can this gap be widened? Is there a language L with a CFG of size $O(n)$ but whose smallest DPDA has size $\Omega(2^{2^{2^n}})$? We show yes (sort of). And, in fact, you can replace $2^{2^{2^n}}$ by any computable function (sort of). Actually you can replace the phrase *any computable function* with an even large class of functions; however, we will get to that later. Unfortunately (1) we will prove that for an infinite number of n (as opposed to for all n) there are languages L_n with small CFG's and ginormous DPDA's, (2) the languages L_n will not be explicit.

We will need Turing Machines. We do not define them formally here (the definition is any standard textbook) but we do mention several conventions we will be using.

Def 18.1 Let M be a Turing Machine. A *configuration* (config) of M is a string of the form $\alpha_1 {}^{q}_{\sigma} \alpha_2$ where $\alpha_1, \alpha_2 \in \Sigma^*$, $\sigma \in \Sigma$, and $q \in Q$. We interpret this as saying that the machine is in state q and the head is looking at the square where we put the ${}^{q}_{a}$. Note that our configuration represents what is happening, though in reality the state is NOT on the tape. Intuitively, a config is a picture of all you need to know about the Turing machine to proceed.

Notation 18.2 If $M(y)$ halts we write $M(y) \downarrow$. If $M(y)$ does not halt then we write $M(y) \uparrow$.

Convention 18.3 Let M be a Turing Machine and x be an input. If $M(x) \downarrow$ then we represent the computation by a sequence of config's. The first config will be how the machine starts with x on the input. The last config will be in an ACCEPT state. Each config follows from the last one via the TM's instructions. Our convention is that all of the config's are the same size. We can assume this since, if $M(x) \downarrow$ then there is a finite number L such that the computation $M(x)$ used $\le L$ tape squares.

We note two sets that will be useful to us.

- $HALT = \{(x, e) : M_e(x) \downarrow\}$.
- $INF = \{e : \text{ there are an infinite number of } x \text{ such that } M_e(x) \downarrow\}$.

Def 18.4 A set is *computably enumerable*[1] (henceforth c.e.) if there is a Turing machine M such that $x \in A$ iff $M(x) \downarrow$.

We will need the following well known facts.

Lemma 18.5

[1]This concept used to be called *recursively enumerable* ; however, the logic community has switched to computably enumerable. See Soare's article [9] for a historical and philosophical discussion of the change.

1. INF is not c.e.

2. If B is a decidable set then any question of the form $(\exists x)[B(x) = 1]$ can be phrased as a query to $HALT$.

3. If B is a decidable set then any question of the form $(\exists x)(\forall y)[B(x, y) = 1]$ can be phrased as a query to INF.

18.1 An Interesting CFG

Def 18.6 Let $M_1, M_2, \ldots,$ be a standard list of Turing machines over the alphabet Σ. Let \$ be a symbol that is not in Σ. We assume that any halting computation takes an even number of steps. For $e \in \mathsf{N}$ let ACC_e be the set of all sequences of config's represented by

$$\$C_1\$C_2^R\$C_3\$C_4^R\$\cdots\$C_L^R\$$$

such that

- $|C_1| = |C_2| = \cdots = |C_L|$.
- The sequence $C_1, C_2, \ldots, C_L$ represents an accepting computation of M_e on some input.

As usual $\overline{ACC_e}$ is the compliment of ACC_e.

Lemma 18.7 *For all e, $\overline{ACC_e}$ is a CFL.*

Proof: We give two proofs. The second one requires knowing that CFG's are equivalent to PDA's.

1) Assume that the machine has start state s and halt states h_{accept} and h_{reject}. Any element of ACC_M must have one s and one h_{accept} and no h_{reject}. Recall that $\$ \notin \Sigma^*$.

When is a string SEQ NOT in ACC_M? We give a partial list.

1. SEQ begins $\$y\$$ where y is not in the right format to be a starting config.
2. SEQ ends $\$y\$$ where y is not in the right format to be an accepting config.
3. SEQ has either 0 or at least 2 symbols in $\Sigma \times \{s\}$.
4. SEQ has either 0 or at least 2 symbols in $\Sigma \times \{h_{accept}\}$.
5. SEQ has either at least 1 symbols in $\Sigma \times \{h_{reject}\}$.
6. SEQ has as a subword an element of $\{\$y\$z\$: y, z \in \Sigma^*, |y| \neq |z|\}$.

All of the above are CFL's.

If SEQ is not in any of the above sets then it must be of the form

$$\$C_1\$C_2^R\$C_3\$C_4^R\$ \cdots \$C_L^R\$$$

where

- C_1 represents a starting configuration with an input x.
- Each C_i is of the same length.
- C_L is an accepting configuration.

We now need to write a CFG that says that there is some i such that C_{i+1} does not follow from C_i. There are two ways this can happen: the wrong move is made or the contents of the tape change when the move of the Turing machine shouldn't change it (e.g., a tape square that is not that close to the head gets changed.) We do an example.

Lets say that one of the instructions is $\delta(q, a) = (p, L)$ (if the machine is in state q and the head is looking at an a then the state changes to p and the head moves left). So we want to make sure that if C_i has b_a^q then C_{i+1} won't have ${}_{\sigma}^{p}, b$ in the right place. Let X be all ordered pairs that are not ${}_{\sigma}^{p}, b$.

$S \rightarrow TUT$

$T \rightarrow T\sigma$ for all σ

$T \rightarrow e$

$U \rightarrow \sigma U\sigma$ for all σ

$U \rightarrow V$

$V \rightarrow p_a^q W\tau_1\tau_2$ for all $\tau_1\tau_2 \in X$

$V \rightarrow \sigma V\sigma$ for all σ

$V \rightarrow e$

If $S \rightarrow \alpha$ and α is not already one of the strings generated by one of the above cases then S is a sequence of the right form, but there is a C_i and a C_{i+1} such that C_{i+1} does not follow from C_i because the instruction was not carried out properly. What we just did for the one instruction we do for all of the instructions.

We also need to generate the sequences where the instructions are carried out at the head but someplace else changes when it shouldn't. We leave this to the reader.

In the end we have a finite number of CFG's. We take their union to get the CFG we need.

2) We describe a PDA for $\overline{ACC_e}$. The set of inputs that are not of the form

$$\$C_1\$C_2^R\$C_3\$C_4^R\$\cdots\$C_L^R\$$$

is clearly PDA-recognizable. We need only describe a PDA for strings of this form which don't represent an accepting computation. When a C_i comes in put it on the stack. When you read C_{i+1} (backwards) pop off the stack seeing that they match until you get to the place where they should differ. If you ever see a place where they differ in a way that is not how the Turing machine says they should, then accept.

▮

Lemma 18.8 *Let $e \in \mathsf{N}$.*

1. *If $e \in INF$ then $\overline{ACC_e}$ is not a DCFL.*

2. *If $e \notin INF$ then $\overline{ACC_e}$ is regular, and hence is a DCFL.*

Proof:

If $e \in INF$ then ACC_e has an infinite number of strings of the form

$$\$C_1\$C_2^R\$C_3\$C_4^R\$\cdots\$C_L^R\$$$

where $|C_1| = |C_2| = \cdots = |C_L|$. By an easy application of the pumping lemma for CFL's, ACC_e is not a CFL. Since DCFG's are closed under complementation, $\overline{ACC_e}$ is not a DCFG.

If $e \notin INF$ then ACC_e is finite, hence regular. Since regular languages are closed under complementation, $\overline{ACC_e}$ is regular. ▮

18.2 Small CFG, Ginormous DPDA

The following theorem is due to Hartmanis [4].

Theorem 18.9 *Let f be any computable function.*

1. *f is not a bounding function for (DPDA,PDA).*

2. *For an infinite number of n there is a language L_n such that*

   (a) Any DPDA for L_n has size $\ge f(n)$.

   (b) There is a CFG for L_n of size $\le n$.

(Part 2 follows from part 1 so we will not prove it.)

Proof:

Assume, by way of contradiction, that there is a computable bounding function for (DPDA,PDA). We use this to show that INF is c.e., contrary to Lemma 18.5.1.

We present an algorithm that halts on e iff $e \in INF$.

ALGORITHM A

1. Input(e)
2. Construct the PDA P for $\overline{ACC_e}$.
3. Compute $N = f(|P|)$. Let $D_1, \ldots, D_s$ be all of the DPDA's of size $\le N$.
4. Search for $(x_1, \ldots, x_s)$ such that $(\forall i)[P(i) \neq D_i(x_i)]$. If ever such is found then stop.

END OF ALGORITHM

$e \in INF \implies \overline{ACC_e}$ is not DCFL $\implies (\forall i \le s)(\exists x_i)[P(x_i) \neq D_i(x_i)] \implies A(x)\downarrow$

$e \notin INF \implies \overline{ACC_e}$ is DCFL $\implies (\exists i \le s)(\forall x)[P(x) = D_i(x)] \implies A(x)\uparrow$

▮

Open Problem 18.10 Let f be a computable function. Does there exists, for all n, a language L_n such that

1. Any DPDA for L_n has size $\ge f(n)$.
2. There is a CFG for L_n of size $\le n$.

By Corollary 17.2 we have this for $f(n) = 2^{2^{\Omega((\sqrt{n})}}$. Can we obtain it for faster growing f? Can we obtain a proof where the languages are more concrete then the ones in Theorem 18.9.

There is no computable bounding function for (DPDA,PDA). Can we say more about this? We can but we need some more concepts.

Def 18.11

1. An *oracle Turing Machine*, denoted $M^{()}$, is a Turing machine that has the ability to make queries to an oracle. The oracle is a set and the queries are Boolean. The statement $M^A(x)$ means that you run oracle Turing machine M with oracle A, on input x. A formal definition would involve being able to write the query on a separate oracle tape and (magically!) get the answer. We leave the details to the reader.

2. $A \leq_{\mathrm{T}} B$ if there exists an oracle Turing machine $M^{()}$ such that A is decided by M^B. We also say that A is *computable-in-B*.

Def 18.12 A set is *computably enumerable in B* (henceforth *c.e.-in-B*) if there is an oracle Turing machine $M^{()}$ such that $x \in A$ iff $M^B(x) \downarrow$.

We will need the following well known fact.

Lemma 18.13 *INF is not c.e. in $HALT$.*

The following theorem is due to Hay [5]. Given what we've done so far we have the following porism[2] of Theorem 18.9 given Lemma 18.13.

Porism 18.14 Let f be any computable-in-$HALT$ function. Then f is not a bounding function for (DPDA,PDA).

Can we characterize how fast f can grow to be a bounding function for (DPDA,PDA)? We can. We show that if (1) if f is a bounding function for (DPDA,PDA) then $INF \leq_{\mathrm{T}} f$, and (2) there is a bounding function f for (DPDA,PDA) such that $f \leq_{\mathrm{T}} INF$. The first result appears to be new. The second one was proven by Hay [5].

We need a weaker theorem first.

[2] A *porism* to a Theorem T is a statement whose proof can be obtained by obvious changes to the proof of Theorem T.

Theorem 18.15 *If f is a bounding function for (DPDA,PDA) then $HALT \le_{\mathrm{T}} f$.*

Proof:

Let $ACC_{e,x}$ be the set of sequences of accepting configurations that represent the computation $M_e(x) \downarrow$. Note that

- If $M_e(x) \downarrow$ then $ACC_{e,x}$ has one string, which is the accepting computation of $M_e(x)$.
- If $M_e(x) \uparrow$ then $ACC_{e,x} = \emptyset$.
- For all e, x the language $ACC_{e,x}$ is regular.
- Given e, x one can construct a PDA for $\overline{ACC_{e,x}}$ by Lemma 18.7.

ALGORITHM FOR $HALT$ THAT USES f

1. Input(e, x)
2. Construct the PDA P for $\overline{ACC_{e,x}}$.
3. Compute $N = f(|P|)$. Let $D_1, \ldots, D_s$ be all of the DPDA's of size $\le N$. Create the DPDA's for their complements, which we denote $E_1, \ldots, E_s$.
4. Apply the algorithm from Theorem 13.1 to each element of $E_1, \ldots, E_s$. This will produce a set of $t \le s$ DFA's, $F_1, \ldots, F_t$ such that all of the languages in $\{L(E_1), \ldots, L(E_s)\}$ that are regular are represented by some element of $\{F_1, \ldots, F_t\}$.
5. For each $1 \le i \le t$ determine if F_i determine recognizes exactly one element. If it does then determine if that element is an accepting computation of $M_e(x)$. If there is such an i then output YES. If not then output NO.

END OF ALGORITHM

Assume $(e, x) \in HALT$. Then $ACC_{e,x}$ has one element and that element is an accepting computation of $M_e(x)$. By the definition of f $(\exists i \leq s)[L(D_i) = \overline{ACC_{e,x}}]$, hence $(\exists i \leq t)[L(E_i) = ACC_{e,x}]$. Since $ACC_{e,x}$ is finite (it has either 0 or 1 strings) it is regular. Hence $(\exists i \leq t)[L(F_i) = ACC_{e,x}]$. The algorithm will find this F_i and output YES.

Assume the algorithm outputs YES. Then an accepting computation for $M_e(x)$ has been found. Therefore $(e, x) \in HALT$. ▮

Theorem 18.16

1. *If f is a bounding function for (DPDA,PDA) then $INF \leq_{\mathrm{T}} f$.*

2. *Let f be such that $INF \not\leq_{\mathrm{T}} f$. For an infinite number of n there is a language L_n such that*

   (a) Any DPDA for L_n has size $\geq f(n)$.

   (b) There is a CFG for L_n of size $\leq n$.

   (Part 2 follows from part 1 so we will not prove it.)

Proof:

We freely use Lemma 18.5.2 to phrase $\exists$-questions as queries to $HALT$, and Theorem 18.15 to phrase questions to $HALT$ as calls to f.

ALGORITHM FOR INF THAT USES f

1. Input(e)

2. Construct the PDA P for $\overline{ACC_e}$.

3. Compute $N = f(|P|)$. Let $D_1, \ldots, D_s$ be all of the DPDA's of size $\leq N$.

4. Ask
$$(\exists x_1, \ldots, x_s)(\forall i \le s)[P(i) \ne D_i(x_i)].$$
If YES then output YES. If NO then output NO.

END OF ALGORITHM

$e \in INF \implies \overline{ACC_e}$ is not DCFL $\implies (\exists x_1, \ldots, x_s)(\forall i \le s)[P(x_i) \ne D_i(x_i)] \implies A(x) = YES$.

$e \notin INF \implies \overline{ACC_e}$ is DCFL $\implies (\exists i \le s)(\forall x)[P(x) = D_i(x)] \implies A(x) = NO$ ▮

Now that we have a lower bound on the complexity of f we turn to upper bounds.

Theorem 18.17 *There exists a bounding function f for (DPDA,PDA) such that f is computable in INF.*

Proof:

In the algorithm below we freely use Lemma 18.5.4 to phrase $(\exists)(\forall)$-questions as queries to INF.

Algorithm for f

1. Input(n)
2. MAX=0.
3. For every PDA P of size $\le n$ do the following
   (a) Ask $(\exists$DPDA $D)(\forall x)[P(x) = D(x)]$?
   (b) If YES then for $i = 1, 2, 3, \ldots$ ask $(\exists$DPDA $D, |D| = i)(\forall x)[P(x) = D(x)]$?
   until the answer is YES.
   (c) Let i be the value of i when the last step stopped. Note that $(\exists D, |D| = i)(\forall x)[P(x) = D(x)]$. If $i > MAX$ then $MAX = i$.

4. Output MAX.

▮

To summarize:

- If f is a bounding function for (DPDA,PDA) then $INF \leq_{\rm T} f$.
- There exists a bounding function for (DPDA,PDA) such that $f \leq_{\rm T} INF$.

Another way of saying this is that the Turing degree of the bounding function is at the second level of the arithmetic hierarchy. This is both an upper and lower bound, and they match.

19 Small CSG for L, Large CFG for $\overline{L}$

It is known that CFL's are not closed under complementation. Is there a language L such that L and $\overline{L}$ are both CFL's but (1) there is a small CFG for L, and (2) any CFG for $\overline{L}$ is large. Yes.

Theorem 19.1 *For all n there exists L_n such that*

1. *Any CFG for $\overline{L_n}$ is of size $\Omega(\frac{2^{n/2}}{n})$.*
2. *There is a CSG for L_n of size $O(\log n)$.*

Proof:

Let $L_n = \overline{\{ww : |w| = n\}}$.

By Theorem 20.11 any CFG for L_n is of size $\Omega(\frac{2^{n/2}}{n})$. By the proof of Theorem 17.1 there is a CFG for L_n of size $O(\log n)$. ▮

Corollary 19.2 *For all n there exists L_n such that*

1. *Any CFG for $\overline{L_n}$ has size $2^{2^{\Omega(\sqrt{n})}}$.*
2. *There is a CFG for L_n of size $O(n)$.*

In Theorem 19.1 we obtained a double exponential gap between the CFG for a language and the CFG for its complement. Can this gap be widened? Is there a language L with a CFG of size $O(n)$ such that the smallest CFG for $\overline{L}$ has $\Omega(2^{2^{2^n}})$? We show yes (sort of). Actually the gap is far worse than that. Unfortunately the proof we give will only show that such languages exist. We do not quite have one explicitly. And we don't get this for all n, just for an infinite number of n.

The proof below is essentially the same as the proof of Theorem 18.9.

Theorem 19.3 *Let f be any computable-in-$HALT$ function. Then there exists infinitely many n such that there exists L_n such that*

1. *Any CFG for $\overline{L}$ has size $f(n)$.*
2. *There is a CFG for L of size n.*

20 Small CSG, Large CFG

In this section we give three examples, of a CFL L_n whose CFG has to be large, but whose CSG is small. The first one is due to Ellul et al [2]. The other two are from Filmus [3], though he proves a very general theorem from which the results fall out, whereas we just prove them from first principles, albeit using essentially a de-generalization of his proof.

Def 20.1 Let G be a CFG and A be a nonterminal of G

1. $GEN(A) = \{w : A \Rightarrow w\}$.
2. Let $w \in L_n$ and let T be a the parse tree for $w \in L(G)$ that contains A. Then $LE(A)$ is the set of leaves that are in the tree below A.

Lemma 20.2 *Let G be a CFG and $w \in L(G) \cap \Sigma^n$. Let $0 < \delta \le 1/2$. There exists $(A, u, v, x) \in N \times \Sigma^* \times \Sigma^* \times \Sigma^*$ such that $w = uvx$, $v \in GEN(A)$, and $\delta n \le |v| \le 2\delta n$.*

Proof: Look at the parse tree for w. Since G is in Chomsky Normal Form the parse tree is binary. Start at the root. At every decision point goto the side that has the most leaves. Let B be the label on the first node such that the $LE(B) \le \delta n$. Let A be the parent of B. A has two children B and C. Note that $LE(A)$ has more than δn nodes below it since B is the first node that has $LE(B) \le \delta n$ nodes below it. Also note that since $LE(B) \le \delta n$ and $LE(C) \le LE(B)$, $LE(C) \le \delta n$. Hence $LE(A) = LE(B) + LE(C) \le 2\delta n$. Hence $\delta n \le LE(A) \le 2\delta n$. Let v be the word generated by A in this parse. Clearly $\delta n \le |v| \le 2\delta n$.

▮

Convention 20.3 If G is a CFG and A is a nonterminal then we assume that A is used in some derivation of an element of $L(G)$.

20.1 Small CSG, Large CFG for $PERM(n)$

Def 20.4

1. If F is a finite set then $PERM(F)$ is the set of all permutations of elements of F. Note that $PERM(F)$ has $|F|!$ elements.
2. For this section $L_n = PERM([n])$.

Lemma 20.5 *Let* $0 < \beta < 1$. *Then* $\frac{n!}{(\beta n)!((1-\beta)n)!} = \Theta\left(\frac{1}{\sqrt{n}}\left(\frac{1}{(1-\beta)^{1-\beta}\beta^{\beta}}\right)^n\right)$

Proof:

By Stirling's Formula $n! \sim \sqrt{2\pi n}(\frac{n}{e})^n$. We use this in the form $n! = \Theta(\sqrt{n}(\frac{n}{e})^n)$. We omit the symbol Θ in our calculations.

$$(\beta n)!(((1-\beta)n))! \sim \sqrt{\beta n}\left(\frac{\beta n}{e}\right)^{\beta n}\sqrt{((1-\beta)n)}\left(\frac{(1-\beta n)}{e}\right)^{(1-\beta)n} =$$

$$\frac{(\sqrt{\beta(1-\beta)})n}{e^n}(\beta n)^{\beta n}((1-\beta)n)^{(1-\beta)n} = \frac{(\sqrt{\beta(1-\beta)})n}{e^n}((1-\beta)n)^n\big(\frac{\beta}{1-\beta}\big)^{\beta n}$$

Inverting this and multiplying by $\sqrt{n}(\frac{n}{e})^n$ yields

$$\sqrt{n}\left(\frac{n}{e}\right)^n \frac{e^n}{\sqrt{\beta(1-\beta)n}}\frac{1}{((1-\beta)n)^n}\left(\frac{1-\beta}{\beta}\right)^{\beta n} = \frac{1}{\sqrt{n}}\frac{1}{(1-\beta)^n}\left(\frac{1-\beta}{\beta}\right)^{\beta n} =$$

$$\frac{1}{\sqrt{n}}\left(\frac{(1-\beta)^{\beta-1}}{\beta^\beta}\right)^n = \frac{1}{\sqrt{n}}\left(\frac{1}{(1-\beta)^{1-\beta}\beta^\beta}\right)^n$$

∎

Def 20.6 If $n \in \mathsf{N}$ then $[n] = \{1, \ldots, n\}$

Theorem 20.7 *For all n:*

1. *Any Chomsky Normal Form CFG for L_n requires $\Omega\big(\frac{1.89^n}{n^{3/2}}\big)$ nonterminals.*
2. *There is a CSG for L_n that has $O(n^2)$ nonterminals.*

Proof:

1) Let $G = (N, \Sigma, S, P)$ be a Chomsky Normal Form Grammar for L_n. We show that $|N| = \Omega\big(\frac{1.89^n}{n^{3/2}}\big)$.

Claim 1: For all nonterminals A there exists a set $F(A) \subseteq [n]$ such that $GEN(A) \subseteq PERM(F(A))$. (We will use the notation $F(A)$ later in this proof.)

Proof of Claim 1:

Let $v, v' \in GEN(A)$. Then there exists u, x, u', x' such that

- $S \Rightarrow uAx \Rightarrow uvx \in PERM([n])$

- $S \Rightarrow u'Ax' \Rightarrow u'v'x' \in PERM([n])$.

Clearly we have

$$S \Rightarrow u'vx' \in PERM([n]).$$

Since $uvx, uv'x \in PERM([n])$, v and v' must contain exactly the same letters (though they may be in a different order). Let $F(A)$ be the set of letters in v. Clearly $GEN(A) \subseteq PERM(F(A))$.

End of Proof of Claim 1

We map L_n to $N \times [n]$. Given $w \in L_n$ find (A, u, v, x) as in Lemma 20.2 (with $\delta = 1/3$). Let $i = |u| + 1$, so i is where the v-part starts. Map w to (A, i).

We upper bound the size of the inverse image of any $(A, i) \in N \times [n]$ and then use that to lower bound $|N|$.

Let $(A, i) \in N \times [n]$. How many w can map to it? Let $w = uvx$ where v begins at the ith spot and $\frac{n}{3} \leq |v| \leq \frac{2n}{3}$. Note that all of the w's that map to (A, i) have the same $|v|$, namely $|F(A)|$. We denote this by r and note that $\frac{n}{3} \leq r \leq \frac{2n}{3}$.

$v \in PERM(F(A))$. There are at most $r!$ such v. $ux \in PERM(\Sigma - F(A))$. There are at most $(n-r)!$ such ux. Hence there are at most $r!(n-r)!$ elements of L_n that map to (A, i). This is maximized when $r = n/3$ (or $r = 2n/3$). So each element of $N \times [n]$ has at most $(n/3)!(2n/3)!$ elements in the inverse image. Hence we get

$$n! \leq \sum_{(A,i) \in N \times [n]} (n/3)!(2n/3)! \leq |N| n (n/3)!(2n/3)!$$

Therefore, using Lemma 20.5

$$|N| \geq \frac{1}{n} \frac{n!}{(n/3)!(2n/3)!} \geq \Omega\left(\frac{1}{n} \frac{1}{\sqrt{n}} \frac{1}{(1/3)^{1/3}(2/3)^{2/3}}\right) \geq \Theta\left(\frac{1.89^n}{n^{3/2}}\right).$$

2) We give a CSG for L_n that has $O(n^2)$ nonterminals.

$$\begin{aligned}
S &\rightarrow A_1A_2\cdots A_n \\
A_iA_j &\rightarrow A_jA_i \text{ for all } 1 \le i < j \le n \\
A_1 &\rightarrow 1 \\
A_2 &\rightarrow 2 \\
&\vdots \\
A_n &\rightarrow n
\end{aligned}$$

This CSG is of size $\Theta(n^2)$ but is not in Chomsky Normal Form; however, it is easy to convert it to Chomsky Normal Form while keeping the size at $\Theta(n^2)$. ▮

20.2 Small CSG, Large CFG for $\{w : \#_1(w) = \cdots = \#_n(w) = k\}$

Def 20.8 Let $n, k \in \mathsf{N}$ such that $n \equiv 0 \pmod{k}$. Let $\Sigma = [n]$. Let

$$L_{k,n} = \{w : \#_1(w) = \cdots \#_n(w) = k\}$$

Note that every string in $L_{k,n}$ is of length kn.

Theorem 20.9 *For all n:*

1. *Any Chomsky Normal Form CFG for $L_{k,n}$ requires $\Omega\left(\frac{BILL-FILLIN}{LATER}\right)$ nonterminals.*
2. *There is a CSG for $L_{k,n}$ that has $O(kn + n^2)$ nonterminals.*

Proof:

1) Let $G = (N, \Sigma, S, P)$ be a Chomsky Normal Form Grammar for $L_{k,n}$. We show that $|N| = \Omega\left(\frac{BILL-FILLIN}{LATER}\right)$.

Claim 1: For all nonterminals A there exists $j_1, \ldots, j_n \in \{0, \ldots, n\}$ such that

$$GEN(A) = \{w : (\forall i)[\#_i(w) = j_i]\}.$$

Proof of Claim 1:

Let $v, v' \in GEN(A)$. Then there exists u, x, u', x' such that

- $S \Rightarrow uAx \Rightarrow uvx \in L_{k,n}$
- $S \Rightarrow u'Ax' \Rightarrow u'v'x' \in L_{k,n}$

Clearly we also have

$$S \Rightarrow u'vx' \in L_{k,n}.$$

Hence we must have $(\forall i)[\#_i(v) = \#_i(v')]$. Clearly the claim holds.

End of Proof of Claim 1

Def 20.10 If A is a nonterminal then let $F(A)$ be the $(j_1, \ldots, j_n)$ proven to exist in the above claim.

Let N be the set of nonterminals of G. We map $L_{k,n}$ to $N \times [n]$. Given $w \in L_{k,n}$ find (A, u, v, x) as in Lemma 20.2 (with $\delta = 1/3$). Let $i = |u| + 1$, so i is where the v-part starts. Note that $\frac{kn}{3} \leq |v| \leq \frac{2kn}{3}$. Map w to (A, i).

We upper bound the size of the inverse image of any $(A, i) \in N \times [n]$ and then use that to lower bound $|N|$. We need a definition: if $v \in \Sigma^*$ (likely not an element of $L_{k,n}$) then the *signature of* v is the tuple $(\#_1(v), \ldots, \#_n(v))$.

Let $(A, i) \in N \times [n]$. How many w can map to it? Let $w = uvx$ where v begins at the ith spot and $\frac{kn}{3} \leq |v| \leq \frac{2kn}{3}$. Note that all of the w's that map to (A, i) have the same signature $(j_1, \ldots, , j_n)$. Note $\frac{kn}{3} \leq \sum_{i=1}^{n} j_i \leq \frac{2kn}{3}$.

Let $\sum_{i=1}^{n} j_i = j$. How many strings are there of the form uvx are there such that (1) $uvw \in L_{k,n}$, (2) v is of length j, (3) v has signature $(j_1, \ldots, j_n)$. There are $\frac{j!}{\prod_{i=1}^{n} j_i!}$ ways to pick v. Then there are $\frac{(kn-j)!}{\prod_{i=1}^{n}(k-j_i)!}$ ways to pick ux. Hence the number of such uvx is

$$\frac{j!}{\prod_{i=1}^{n} j_i!} \frac{(kn-j)!}{(k-j_i)!} \leq$$

This is maximized when $j = \frac{kn}{3}$ and $j_i = \frac{j}{n} = \frac{k}{n}$. (We assume 3 divides k.) Hence the maximum number of w that map to (A, i) is

$$\frac{\frac{kn}{3}!}{(\frac{k}{n})^n} \frac{\frac{2kn}{3}!}{\frac{2k}{3}!}$$

The number of elements in $L_{k,n}$ is $\frac{(kn)!}{(k!)^n}$. Hence we have

$$\frac{(kn)!}{(k!)^n} \leq N \times \frac{\frac{kn}{3}!}{(\frac{k}{n})^n} \frac{\frac{2kn}{3}!}{\frac{2k}{3}!}$$

Hence

$$N \geq \frac{(kn)!}{\frac{kn}{3}!\frac{2kn}{3}!} \frac{(\frac{k}{n})^n \frac{2k}{3}!}{(k!)^n}$$

BILL- NEED TO FINISH LATER- MIGHT GIVE UP ENTIRELY.

2) We give a CSG for $L_{k,n}$ that has $O(kn + n^2)$ nonterminals.

$$\begin{aligned} S &\rightarrow A_1^k A_2^k \cdots A_n^k \\ A_i A_j &\rightarrow A_j A_i \text{ for all } 1 \leq i, j \leq n \\ A_i &\rightarrow i \text{ for all } 1 \leq i \leq n \end{aligned}$$

This CSG is not in Chomsky Normal Form; however, it is easy to convert it to such without changing the number of nonterminals by too much. We leave it as an exercise that there will be

$O(kn + n^2)$ nonterminals. ▮

20.3 Small CSG, Very Large CFG for $\{ww : |w| = n\}$

For this section Σ has t letters in it.

$$L_n = \{ww : |w| = n\}.$$

Theorem 20.11 *For all n:*

1. *Any CFG for L_n has size $\Omega(\frac{t^{n/2}}{n})$.*
2. *There is a CSG for L_n of size $O(t \log n)$.*

Proof:

1) Let $G = (N, \Sigma, S, P)$ be a Chomsky Normal Form Grammar for L_n. We show that $|N| = \Omega(2^n)$.

Claim 1:

i) For all nonterminals A there exists m such that $GEN(A) \subseteq \Sigma^m$.

ii) If $m \leq n$ then $|GEN(A)| = 1$

Proof of Claim 1:

i) Let $v, v' \in GEN(A)$. Then there exists u, x, u', x' such that

- $S \Rightarrow uAx \Rightarrow uvx \in L_n$
- $S \Rightarrow u'Ax' \Rightarrow u'v'x' \in L_n$

Clearly we also have

$$S \Rightarrow uv'x \in L_n$$

Since all elements of L_n are of length $2n$ we have the following:

- $|uvx| = 2n$, so $|v| = 2n - |ux|$.
- $|uv'x| = 2n$, so $|v'| = 2n - |ux|$.
- Let $i_1 = |v| = |v'|$. Let $i_2 = |u|$.

ii) We show that if $i_1 \le n$ then $v = v'$. Since $uvx \in L_n$ and $uv'x \in L$, there exists w, w' such that $uvx = ww$ and $uv'x = w'w'$.

Let $w = \sigma_1\sigma_2 \cdots \sigma_n$ and $w' = \sigma_1'\sigma_2' \cdots \sigma_n'$

Case 1: $v = \sigma_i \cdots \sigma_j$ and is contained in the first w. Hence $x = \sigma_{j+1} \cdots \sigma_n \sigma_1 \cdots \sigma_n$. Since $w'w' = uv'x$ and $ww = uvx$, we have $w = w'$ so $v = v'$.

Case 2: $v = \sigma_i \cdots \sigma_j$ and is contained in the second w. This is similar to Case 1.

Case 3: $v = \sigma_i \cdots \sigma_n \sigma_1 \cdots \sigma_j$ and $i \le j$. This cannot occur since then $|v| \ge n + 1$.

Case 4: $v = \sigma_i \cdots \sigma_n \sigma_1 \cdots \sigma_j$ and $i > j$.

Hence (1) $u = \sigma_1 \cdots \sigma_i$ and (2) $v = \sigma_{j+1} \cdots \sigma_n$. Therefore (1) $\sigma_1' = \sigma_1$, …, $\sigma_i' = \sigma_i$ and (2) $\sigma_{j+1}' = \sigma_{j+1}$, …, $\sigma_n' = \sigma_n$. Since $i > j$ we have $\sigma_1' = \sigma_1$, …, $\sigma_n' = \sigma_n$. Hence $w = w'$ and $v = v'$.

End of Proof of Claim 1

Claim 2: Let $|v| \le n$ and let $i \in [n]$. The number of strings of the form $uvx \in L_n$ where $|u| = i$ is $\le t^{n-|v|}$.

Proof of Claim 2:

Let $uxv = \sigma_1 \cdots \sigma_n \sigma_1 \cdots \sigma_n$. Since $|v| \le n$ and i is determined v determines exactly $|v|$ of the σ's. Hence there are just $t^{|v|-n}$ σ's to determine.

End or Proof of Claim 2

Def 20.12 If A is a nonterminal let $F(A)$ be the value of m that exists by Claim 1.1

Let N be the set of nonterminals of G. We map L_n to $N \times [n]$. Given $ww \in L_n$ find (A, u, v, x) as in Lemma 20.2 with $\delta = 1/2$. Map ww to $(A, |u|)$. Note that $\frac{n}{2} \le |v| \le n$ and that the v part of ww starts at position $|u| + 1$.

We upper bound the size of the inverse image of any (A, i) and then use that to lower bound $|N|$. Note that since $v \in GEN(A)$ has length $\le n$, $GEN(A) = \{v\}$. Then any element of L_n that maps to (A, i) is of the form $u'vw'$ where $|u'| = i$. By Claim 2 there are at most $t^{n-|v|} \le t^{n/2}$ such strings. Hence the inverse image of (A, i) has at most $t^{n/2}$ elements. Therefore $t^n \le N \times i \times t^{n/2}$, so $N \ge \frac{t^{n/2}}{n}$.

2) We give a CSG (not in Chomsky Normal Form) for L_n in the $t = 2$ case that has $O(\log n)$ nonterminals and then show how to convert it to one in Chomsky Normal Form of size $O(\log n)$. We leave it to the reader to extend this to the general t case.

We first construct a CSG G for L_n of size $O(n)$ that is not in Chomsky Normal Form. We will then show how to find a CSG G' such that $L(G') = L(G)$ and G' is in Chomsky Normal Form. but G' has size $O(\log n)$.

The nonterminals are S, X, A, B, A', B'. S is the start symbol.

Rules to get started.

$S \to X^{n-2}Aa$

$S \to X^{n-2}Bb$

Rules to put the A' and B' in place.

$X \to aA'$

$X \to bB'$

Rules to move the A' and B' to the right.

$A'a \to aA'$

$A'b \to bA'$

$B'a \to aB'$

$B'b \to bB'$

Rules to move the A or B to the left and at the same time create a' and b's out of A''s and B's.

$A'A \to Aa$

$A'B \to Ba$

$B'A \to Ab$

$B'B \to Bb$

Rules to change A to a and B to b. To be used at the end.

$A \to a$

$B \to b$

We give an example of how to generate $aabaaaabaa$.

Use the rules to get started to obtain

$S \to XXXAa$ (We use Aa since the word $aabaa$ ended in a.)

Use the rules to put the A' and B' in place to obtain

$XXXAa \Rightarrow aA'aA'bB'aA'Aa$

Use the rules to move the A' and B' to obtain

$aA'aA'bB'aA'Aa \Rightarrow aabaA'A'B'A'Aa$

Use the rules to move the A to obtain

$aabaA'A'B'A'Aa \Rightarrow aabaAaabaa \Rightarrow aabaaabaa.$

Inspired by the example the reader can easily prove that any string in L_n can be generated. With a bit more thought the reader can show that this way to derive words is essentially the only way to do it, so the $L(G) = L_n$.

Clearly G has size $O(n)$.

In order to modify G to form G' where $L(G) = L(G')$, G' is in Chomsky Normal Form, and G' is of size $O(\log n)$ use the techniques of Lemma 3.1. ▮

Corollary 20.13 *Let* $|\Sigma| = 2$. *Let* $W_n = L_{2^n}$. *For all* n*:*

1. *Any CFG for* W_n *has size* $2^{2^{\Omega(n)}}$.
2. *There is a* CSG *for* W_n *of size* $O(n)$.

21 Languages that have a Small CSG and Require a Ginormous CFG

In Corollary 20.13 we showed that the language $W_n = \{ww : |w| = 2^n\}$ has a CSG of size $O(n)$ but any CFG for it has size $2^{2^{\Omega(n)}}$. Can we obtain a bigger gap between the size of a CSG and CSF? Yes. Meyer and Fisher [7] say the following in their **Further Results** Section:

. . . context-sensitive grammars may be arbitrarily more succint than context-free grammars . . .

The reference given was a paper of Meyer [6]. That paper seems to only talk about Turing Machines; however, after conversing with Meyer about it, we can now present the proof. This is essentially his proof but scaled down to CSL's and CFG's. We first present a weak version of his theorem, and then the full version.

We state the theorems in terms of CFG's and CSL's but we prove them using PDA's and LBA's. Since the size bounds of CFG's (CSG's) and PDA's (LBA's) are computably-related (in fact polynomial) this will not matter. Let $P_1, P_2, \ldots$ be an easily accessible list of all PDA's. They are in order of size. We assume that P_e is of size $\geq e$.

Theorem 21.1 *Let f be a computable function. For all n there exists L_n such that*

1. *any CFG for L_n is of size at least $f(n)$,*
2. *there is a CSL for L_n of size $O(n)$.*

Proof: We construct the language L_n by describing an $NSPACE(|x|)$ algorithm [3]. The idea is that L_n will diagonalize against all small PDA's.

ALGORITHM for L_n

1. Input(x) (Note that all but the last step depend only on n which is a constant. The last step depends on x.)
2. Compute $f(n)$.
3. For $1 \le i \le f(n)$ deterministically simulate P_i on 1^i.
4. For $a = f(n)+1, f(n)+2, \ldots$ determine if $(\forall i \le f(n))[P_a(1^i) \ne P_i(1^i)]$. Stop when you find such an a. (Such and a will be found since the constraints on P_a only affect a finite number of values.)
5. Simulate nondeterministically the PDA for P_a on x.

For all inputs x the same a is found. Hence $L_n = L(P_a)$. By construction there is no PDA for $L(P_a)$ of size $\le f(n)$.

We now consider the space used on input x. Note that since a is constant (relative to $|x|$) and that every step except the last one requires constant space. The last step is a nondeterministic simulation of a PDA, which can be done in $O(|x|)$ space.

[3]Traditionally n is the length of the input; however, we are using n as a parameter for the size of the machines involved so we cannot use it.

What is the size of the machine described above? The only parameter the machine needs is n. All the rest is constant. Hence the machine is of size $n+O(1)$. We can get it to be size $\log n+O(1)$ but we do not need to.

▮

What was it about CSG's and CFG's that the proof of Theorem 21.1 use? Can it be generalized? We consider this point after proving the next theorem which is stronger.

Theorem 21.1 considers the class of computable functions. Meyer's actually considered the class of computable-in-HALT functions.

Def 21.2

1. If $M_e^{()}$ is an oracle Turing machine and A is a set then $M_{e,s}^A(x)$ is the following: Run $M_e^A(x)$ for s steps. If it halts then output whatever it outputs. If not then output DON"T KNOW.

2. $HALT_s = \{(x,e) : \ M_{e,s}(x) \downarrow \}$.

3. If $f \le_{\mathrm{T}} HALT$ via $M_i^{()}$ then f_s is $M_{i,s}^{HALT_s}(x)$. Note that, for all n, $\lim_{s\to\infty} f_s(x) = f(x)$. Since these are all discrete quanities we have that $(\forall n)(\exists s_0)(\forall s \ge s_0)[f_s(n) = f(n)]$.

Theorem 21.3 *Let $f \le_{\mathrm{T}} HALT$. For all n there exists L_n such that*

1. *any CFG for L_n is of size at least $f(n)$,*

2. *there is a CSL for L_n of size $O(n)$.*

Proof: This proof will be similar to (but not identical to) the proof of Theorem 21.1.

We construct the language L_n by describing an $NSPACE(|x|)$ algorithm. The idea is that L_n will diagonalize against all small PDA's. The difference between this proof and the proof of Theorem 21.1 is that in this proof we can only approximate the line between small and large PDA's via $f_s(n)$.

ALGORITHM for L_n

1. Input(x). Let $s = |x|$. Carry out the algorithm given below unless the computation uses more than s space, in which case you stop and just say NO. Note that all steps but the last only depend on s.

2. Compute $f_s(n)$ and store it.

3. Compute L_n on all strings of length $\le \lg\lg s$ and store the results.

4. Simulate deterministically $P_1, P_2, \ldots, P_{f_s(n)}$ on inputs of length $\le \lg\lg s$. Using the membership information about L_n that was calculated in the last step we find some i, z, $1 \le i \le f_s(n)$ (there may not be any) such that $(P_i) \ne L_n$. Let $ACTIVE = \{i_1, \ldots, i_m\}$ be the set of i such that we do not know that $L(P_i) \ne L_n$. Store the list $ACTIVE$.

5. Search for $(a, \{x_i : i \in ACTIVE\})$ such that $a \ge f_s(n)$ and for all $i \in ACTIVE$, $P_i(x_i) \ne P_a(x_i)$. When one is found goto the next step. (Such and a will be found since the constraints on P_a only affect a finite number of values.)

6. Simulate nondeterministically the PDA for P_a on x.

Since $f \le_{\mathrm{T}} HALT$ there exists s_0 such that, for all $s, s' \ge s_0$, $f_s(n) = f_{s'}(n)$. There will exist $s_0' \ge s_0$ such that for all x, $|x| \ge s_0'$, when the algorithm is run on x the same value of a will be found. We *do not* have that $L(P_a) = L_n$. However, we do have that (1) L_n differs from every $L(P_i)$ with $i \le f(n)$ either because i was found to not equal L_n, or because $i \in ACTIVE$ and a witness to the difference with P_a was found, and (2) L_n and $L(P_a)$ differ on a finite number of strings, hence L_n is a CFL.

The machine operates in nondeterministic linear space since for all but the last step we do not allow it to use more than $|x|$ space, and the last step is a nondeterministic simulation of a PDA which is nondeterministic linear space.

The machine is of size n for the same reasons given for the machine in Theorem 21.1. ▮

What was it about CFG's and CSG's that made this proof work? The fact that we could simulate CFG's with CSG's with very little overhead is all we needed. We will not formalize this; however, we will still use it in the next section.

BILL- NEED UPPER BOUNDS.

22 Small TM, Ginormous CSG

There is an enormous size difference between TM's and CSG's. The following porism can be obtained by modifying the proof Theorem 21.3. The theorem is due to Meyer [6].

Porism 22.1 Let $f \leq_{\mathrm{T}} HALT$. For all n there exists L_n such that

1. any CSG for L_n is of size at least $f(n)$,
2. there is a TM for L_n of size $O(n)$.

BILL- NEED UPPER BOUNDS

23 Acknowledgment

We thank: Richard Beigel for help with some of the context free grammars in this paper; Albert Meyer for help with Theorem 21.3; Jefferey Shallit whose paper [2] inspired this survey; Karthik Gopalan for help with some of the proofs and for proofreading;, and Sam Zbarsky for help with Lemma 3.1.

References

[1] A. Ehrenfeucht and P. Zeiger. Complexity measures for regular expressions. *Journal of Computer and System Sciences*, 12(1):134–146, 1976.

[2] K. Ellul, B. Krawetz, J. Shallit, and M. Wang. Regular expressions: new results and open problems. *Journal of Automata, Languages, and Combinatorics*, 10(4):407–437, 2005.

[3] Y. Filmus. Lower bounds for context-free grammars. *Information Processing Letters*, 111(18):895–898, 2011.

[4] J. Hartmanis. On the succinctness of different representations of languages. *SIAM Journal on Computing*, 9(1):114–120, 1980.

[5] L. Hay. On the recursion-theoretic complexity of relative succinctness of representations of languages. *Information and Computation*, 52(1):1–7, 1982.

[6] A. Meyer. Program size in restricted programming languages. *Information and Control*, 21(4):382–394, 1972.

[7] A. Meyer and M. Fischer. Economy of description by automata, grammars and formal systems. In *Proceedings of the 12th Annual Symposium on Switching and Automta Theory*, pages 188–191, Washington, DC, 1971. IEEE.

[8] C. H. Papadimitriou and H. R. Lewis. *Elements of the Theory of Computation*. Prentice-Hall, Inc., 1981.

[9] R. Soare. Computability and recursion. *The Bulletin of Symbolic Logic*, 2(4), 1996.

[10] R. E. Stearns. A regularity test for pushdown machines. *Information and Control*, 11(2):323–340, 1967.

[11] S. Yu, Q. Zhuange, and K. Salomaa. The state complexities of some basic operations on regular languages. *Theoretical Computer Science*, 125(2):315–328, 1994.