SUCCINTNESS OF DESCRIPTIONS OF UNAMBIGUOUS CONTEXT-FREE LANGUAGES*

Erik Meineche Schmidt

TR 76-277

April 1976

Department of Computer Science
Cornell University
Ithaca, New York 14853

*This work was supported in part by Aarhus University, Aarhus, Denmark and by "Thanks to Scandinavia, Inc.", New York.

SUCCINTNESS OF DESCRIPTIONS OF UNAMBIGUOUS CONTEXT-FREE LANGUAGES*

Erik Meineche Schmidt

Abstract:

There is no recursive function bounding the succintness gained using ambiguous grammars over unambiguous ones in the description of unambiguous context-free languages.

* This work was supported in part by Aarhus University, Aarhus, Denmark and by "Thanks to Scandinavia, Inc.", New York.

1. Introduction

In this paper we examine the relationship between the size of ambiguous and unambiguous context-free grammars generating the same language. We show that for certain languages the presence of ambiguity in a grammar allows it to be much smaller than any unambiguous one. Specifically we show that for any recursive function there is a language such that the gap between the size of the two types of grammars is not bounded by the function.

The same result is known in two other cases. Meyer and Fischer [Meyer & Fischer] proved it for descriptions of regular (in fact cofinite) languages using finite automata and ambiguous context-free grammars, and Valiant [Valiant] recently proved it for deterministic pushdown automata and unambiguous grammars. Both results are based on the idea of encoding large Turing Machine computations in small context-free grammars [Hartmanis]. The result in [Meyer & Fisher] follows easily from this whereas Valiant has to prove a "repetition lemma" for dpda's in order to get his theorem.

Here we also use the encoding of TM computations but rather than the dpda property we use Ogdens Lemma for context-free languages [Ogden] in an argument which is similar to the proof of the existence of (inherently) ambiguous languages.

2. Encoding of Turing Machine Computations

In order to talk about economy of descriptions we first have to define what we mean by size of Turing Machines and context-free grammars. We use the same definition as in [Valiant].

Definition

a) Let M be a TM with q states and s tapesymbols. The size of M (size(M)) is equal to $q \cdot t$.

b) Let G be a context-free grammar.[+] It's size (size(G)) equals the total number of occurrences of terminal and nonterminal symbols in the productions.

The Turing Machines we consider is the class (denoted by $\mathcal{B}$) of deterministic, one-tape (one-way infinite) machines which halt when started on blank tape. For technical reasons they are modified in the following way.

1) We add four new states (q,r,t,f) and make q the start state and f the only final state.

2) The instructions are modified such that the machine from q enters its original start state without moving — and such that it when in one of the original final states does the following:

[+]We assume that the grammars contain no useless symbols.

1. rewinds the tape using state r
2. enters state t and marks all used tape-squares with the symbol A
3. goes into state f and halts scanning the first blank square on the tape.

3. We make sure that the machine never prints a blank.

Let $\mathcal{U}$ be the class of modified $\mathcal{B}$-machines. Clearly the $\mathcal{U}$-machines use the same amount of tape as the corresponding $\mathcal{B}$-machines.

Now let M be an arbitrary machine from $\mathcal{U}$. We shall only be interested in the (finite) computation obtained by starting M on blank tape and we shall assume that it is of even length. From now on we refer to it as simply "the computation" or "M's computation". With M we associate the following two languages whose elements are M-configurations pairwise related by M's transition function Next_M.

$$L_1^M = \{\#x_0\#x_1^R\#x_2\#x_3^R\#\ldots\#x_{2n-2}\#x_{2n-1}^R\#x_{2n} \mid n\geq 1,\ x_0 \in qB^*$$
$$\&\ (|x_{2i}|=|x_{2i+1}| \wedge x_{2i+1} = \text{Next}_M(x_{2i}))\ \text{for}\ 0\leq i<n\}$$

$$L_2^M = \{\#y_0\#y_1^R\#y_2\#\ldots\#y_{2n-2}\#y_{2n-1}^R\#y_{2n}\# \mid n\geq 1,\ y_{2n} \in A^*f$$
$$\&\ (|y_{2i+1}|=|y_{2i+2}| \wedge y_{2i+2} = \text{Next}_M(y_{2i+1}))$$
$$\text{for}\ 0\leq i<n\}$$

x^R is the mirror-image of x and B is the blank symbol. Note that x_0 is the start configuration for M's computation (padded with blanks) and that y_{2n} is some halting configuration for M.

It is easy to see that L_1^M and L_2^M have unambiguous context-free grammars G_1^M and G_2^M respectively whose sizes are no bigger than a constant times the size of M. If we take their union we get a grammar G^M for the language

$$L^M = L_1^M \cup L_2^M$$

having the property that

$$\underline{\text{size}}(G^M) \leq c.\underline{\text{size}}(M)$$

for some constant c. This grammar however is ambiguous because $L_1^M \cap L_2^M = \{z\}$ where z represents M's computation $(c_0, c_1, \ldots, c_{2n})$ in the following way.

$$z = \#z_0\#z_1^R\#z_2\#\ldots\#z_{2n-1}^R\#z_{2n}\#$$

where $z_i = c_i\, B^{N+1-|c_i|}$ and N is the amount of tape used during the computation. Note that all the z_i's have the same length.

Since $L_1^M \cap L_2^M$ is finite L^M also has an unambiguous grammar. One way of constructing such a grammer is to first construct a pda P_2^M accepting L_2^M. From P_2^M we construct another pda R_2^M which behaves like P_2^M except it uses its finite control to check that it does not accept z. From R_2^M we construct a grammar G_2^M, and take its union with G_1^M. The resulting grammar

is unambiguous and it generates L^M. Its size however is larger than $|z|$ because the extra states used in R_2^M to "look for" z results in extra nonterminals in G_2^M,.

In the next section we are going to prove that any non-ambiguous grammar for L^M must be big if z is long.

3. A property of unambiguous grammars for L^M

Here we prove that since the word z has inherited two different structures (one from L_1^M and another from L_2^M) it will have two different derivations in any small grammar generating L^M. The proof is very similar to the proof that $\{a^i b^j c^k | i = j \vee j = k\}$ is an ambiguous language which can be found in [Aho & Ullman, p. 205]. It uses the following result from [Ogden].

Lemma (Ogden)

Let G be a context-free grammar with m symbols (terminals and nonterminals), let ℓ be the length of the longest right-handside of the productions and let $k = \max\{3, \ell^{2m+3}\}$. If $z \in L(G)$, $|z| \geq k$ and if k or more positions in z are designated as being "distinguished" then z can be written as uvwxy such that

1) w contains at least one of the distinguished positions

2) either u and v both contain distinguished positions or x and y both contain distinguished positions

3) vwx has at most k distinguished positions

4) there is a nonterminal A such that

$S \overset{*}{\Rightarrow} uAy \overset{*}{\Rightarrow} uvAxy \overset{*}{\Rightarrow} uv^iAx^iy \overset{*}{\Rightarrow} uv^iwx^iy$

for all $i \geq 0$ □

Using Ogden's Lemma we can prove the key lemma of the paper.

Lemma 1

Let M be in $\mathcal{U}$ and let N be the amount of tape used in its computation. Let G be any context-free grammar generating L^M and let k be the constant from Ogdens Lemma.

If $N \geq k! + k$ then G is ambiguous.

Proof: Because of the restrictions on the machines in $\mathcal{U}$ the "computation" looks as follows:

z = #qB...B#b...Bp#...#tA...A#A...Af#

where all groups (strings between #'s) are of length N+1 (state and N tapesymbols). We will show that if $N \geq k! + k$ then z has two different derivations in G (recall that $\{z\} = L_1^M \cap L_2^M$).

Assume that $N = k! + j$ where $N > j \geq k$ and consider the word

$$z' = \#\underbrace{qB\ldots B}_{k!+j}\#\underbrace{B\ldots Bp}_{k!+j}\#\ldots\ldots\#\underbrace{A\ldots\ldots}_{k!+j}\#\underbrace{tA\ldots A}_{j}\#\underbrace{A\ldots\ldots Af}_{j}\#$$

with α spanning the first two groups, α_1 the third, α_2 the fourth and α_3 the fifth.

which is obtained from z by replacing its last two groups by shorter ones. Let $z' = \#\alpha\#\alpha_1\#\alpha_2\#\alpha_3\#$ where the α's are as shown above. z' is in L^M (it is in L_2^M) so if we let the j A's in α_3 be distinguished Ogdens Lemma applies and we can write write z' as

$$z' = uvwxy$$

where u,v,w,x and y have properties 1)-4). We shall show that $|x| = |v| \neq 0$ and that v is in α_2 and x is in α_3. The argument is as follows

1) since w contains a distinguished position xy is a suffix of $\alpha_3\#$.

2) If both u and v contain distinguished positions v and x would both be contained in $\alpha_3\#$. Then the word uv^2wx^2y would be of the form $\#\alpha\#\alpha_1\#\alpha_2\#\alpha_3'\#$ where $|\alpha_1| \neq |\alpha_2|$ and $|\alpha_2| \neq |\alpha_3'|$ but there is no such word in L^M. Hence both x and y must contain distinguished positions and it follows that x is a nonempty string of A's in α_3.

3) If w contains α_2 we could again by pumping get a word in the language such that the length of the last three groups were different. But that implies that v contains something in α_2.

4) If v contains a marker (#) it must contain an even number of them (the length of M's computation is even). Assume that v contains markers and consider the following situation:

$$z' = u \underbrace{\eta \# \alpha_1 \#}_{v} \underbrace{\alpha_2 \#}_{w} \underbrace{\alpha_3 \#}_{xy}$$

Then

$$uv^2wx^2y = u \underbrace{\eta\#\alpha_1\#\ldots}_{v} \underbrace{\eta\#\alpha_1\#}_{v} \underbrace{\alpha_2\#}_{w} \underbrace{\alpha_3'\#}_{x^2y}$$

and we have the same contradiction as in 2). Since the case where v contains $\#\alpha_2\#$ can be treated similarly we conclude that <u>v is contained in α_2</u>.

5) If v contains t or $|v| \neq |x|$ we get a contradiction on the word uwy so <u>v is a string of A's and $|x| = |v| \neq 0$</u>.

Since x contains distinguished positions only we know that $|x| < k$ (vwx contains at most k distinguished positions), which means that $|x|$ divides k!. Now there is a nonterminal X and a derivation of z'

$$S \overset{*}{\Rightarrow} uXy \overset{*}{\Rightarrow} uvXxy \overset{*}{\Rightarrow} uvwxy = z'$$

but then we can repeat the subderivation $X \overset{*}{\Rightarrow} vXx$ $k!/|x|$ times to get z

$$S \overset{*}{\Rightarrow} uXy \overset{*}{\Rightarrow} u\, v^{1+k!/|v|} x^{1+k!/|x|}_{x} y \overset{*}{\Rightarrow} uv^{1+k!/|v|}_{w}\, x^{1+k!/|x|}_{y} = z\ .$$

In order to get the other derivation of z we consider the word

$$z'' = \#\underbrace{qB\ldots B}_{j}\#\underbrace{B\ldots Bp}_{j}\#\underbrace{\ldots B}_{k!+j}\#\ldots\ldots\#\underbrace{tA\ldots A}_{k!+j}\#\underbrace{A\ldots Af}_{k!+j}\#$$

where we have now replaced the first two groups in z with shorter ones. Let $\beta_1 = qB \ldots B$ and $\beta_2 = B \ldots Bp$, where the B's in β_1 are distinguished. Using Ogdens Lemma again we can write z" as

$$z'' = u_1 v_1 w_1 x_1 y_1$$

and arguing exactly as before we can show that v_1 is contained in β_1, x_1 in β_2, $|x_1| = |v_1| \neq 0$ and there is a nonterminal Y such that z is derived in the following way

$$s^* \Rightarrow u_1\ Y\ y_1 \overset{*}{\Rightarrow} u_1 v_1^{1+k!/|v_1|}\ Y\ x_1^{1+k!/|x_1|}\ y_1$$

$$\overset{*}{\Rightarrow} u_1 v_1^{1+k!/|v_1|}\ w_1 x_1^{1+k!/|x_1|}\ y_1 = z.$$

Now it only remains to be shown that these two derivations of z must have different derivation trees. Assume the contrary. Then since X generates A's and Y generates B's no node in the tree labelled X can be a descendant of a node labelled Y and vice versa. Hence there is a terminal word t such that $u_1 Ytxy$

is a sentential form. But then we obtain the derivation

$$S \overset{*}{\Rightarrow} u_1 Y t X y \overset{*}{\Rightarrow} u v_1 w_1 x_1 t v w x y = z'''$$

where

$$z''' = \#\beta_1\#\beta_2\#\underbrace{\ldots\ldots\#}_{N}\ldots\ldots\#\underbrace{\ldots\ldots\#}_{N}\alpha_2\#\alpha_1\#$$

z''' however is not in L^M so we have proved that z has two different derivations in G, which means that G is ambiguous. □

4. The size of an unambiguous grammar for L^M.

We use lemma 1 to show that the size of any unambiguous grammar for L^M must grow with the amount of tape used in M's computation.

Lemma 2

Let M be a machine in $\mathcal{U}$ whose computation uses N tape squares. If G_u^M is an unambiguous grammar generating L^M then

$$\underline{\text{size}}(G_u^M) \geq c[\ell\text{og } \ell\text{og } N]^{1/2}$$

where c is a constant.

Proof: In the following c, c_1, c_2, c_3 are suitable constants. From lemma 1 we know that

$$(\ell^{2m+3})! + \ell^{2m+3} > N$$

from which we get

$$N < c_1 (\ell^{2m}) \ell^{2m}$$

Taking log gives

$$\begin{aligned} \log N &< \log c_1 + \ell^{2m} \log (\ell^{2m}) \\ &\leq c_2 \, \ell^{3m} \\ &\leq c_2 \, (\ell+m)^{3(\ell+m)} \end{aligned}$$

and hence

$$\log \log N \leq c_3 \, (\ell+m)^2 \ .$$

Using the fact (which is easy to verify) that for any grammar G $\underline{\text{size}}(G) \geq \frac{1}{2} (\ell+m)$ we get

$$\underline{\text{size}}(G_u^M) \geq c[\log \log N]^{1/2}$$

which is what we want. □

Now we can easily prove our theorem.

Theorem

There is no recursive function F with the following property.

For all ambiguous context-free grammars G_a generating an unambiguous language there exists an unambiguous grammar G_u generating the same language such that

$$\underline{\text{size}}(G_u) \leq F(\underline{\text{size}}(G_a))$$

Proof:

Assume the contrary and consider some L^M where M is in $\mathcal{U}$. Since L^M has an ambiguous grammar G_a^M such that

$$\underline{\text{size}}(G_a^M) \leq \text{const.} \cdot \underline{\text{size}}(M)$$

we get - using lemma 2 and the function F (which we may assume to be increasing) -

$$c[\log \log N]^{1/2} \leq \underline{size}(G_u^M) \leq F(\underline{size}(G_a^M)) \leq F(\text{const. } \underline{size}(M))$$

where N is the amount of tape used in M's computation. But this means that there is a fixed recursive relation between N and $\underline{size}(M)$ for all machines in $\mathcal{U}$ and this immediately enables us to decide whether an arbitrary TM halts when started on blank tape.

Thus we have a contradiction and we may conclude that the theorem is true. □

5. Conclusion

The result proved in this paper answers one of the questions left open in [Valiant]. If we consider the relative succintness gained using finite automata, deterministic push-down automata, unambiguous context-free grammars and ambiguous context-free grammars we get the following table representing the known relations (also from [Valiant]).

type of language described \ description used	fa	dpda	ucfg	acfg
fa	1	recursive	?	nonrec
dpda		1	nonrec	nonrec
ucfg			1	nonrec
acfg				1

The recursiveness of fa - dpda was proved in [Stearns]. The nature of the relation between fa and ucfg is still open.

Acknowledgement

I wish to thank Leonard Berman for his remarks on the first draft of this paper.

References

Aho, A.V. and J.D. Ullman [1972]. The Theory of Parsing, Translation and Compiling, Vol. 1: Parsing, Prentice-Hall, Englewood Cliffs, N.J.

Hartmanis, J. [1967]. Context-free Languages and Turing Machine Computations. Proc. Sympos. Appl. Math., Vol. 19, Amer. Math. Soc., Providence, R.I., 1967, 42-51.

Meyer, A.R. and M.J. Fisher [1971]. Economy of Description by Automata, Grammars and Formal Systems. 12th Amer. Sympos. on Switching and Automata Theory, 188-191.

Ogden, W. [1968]. A Helpful Result for Proving Inherent Ambiguity. Mathematics Systems Theory 2:3, 191-194.

Stearns, R.E. [1967]. A Regularity Test for Pushdown Machines. Information and Control 11, 323-340.

Valiant, L.G. [1975]. A Succinctness Result for Descriptions of Deterministic Languages. TR 70, Centre for Computer Structures, University of Leeds.