INFORMATION AND CONTROL 32, 139–145 (1976)

A Note on the Succinctness of Descriptions of Deterministic Languages

LESLIE G. VALIANT

Centre for Computer Studies, University of Leeds, Leeds, United Kingdom

It ıs shown that the relative succınctness that may be achieved by descrıbing deterministic context-free languages by general unambiguous grammars rather than by deterministic pushdown automata is not bounded by any recursive function.

1. INTRODUCTION

The result proved in this paper is that for the elements of some infinite class of deterministic context-free languages the size of deterministic pushdown automata needed to describe them is not recursively bounded by the size of the smallest unambiguous context-free grammars that generate them. This is a quantitative explanation of the fact that some languages require large descriptions in terms of LR(1) grammars (Knuth, 1965; Aho and Ullman, 1972), or strict deterministic grammars (Harrison and Havel, 1973), even though they can be described very succinctly in terms of general, even unambiguous, context-free grammars. It therefore illustrates one of the tangible advantages of using parsing mechanisms more powerful than a single pushdown stack (e.g., Earley, 1970) even for languages for which that may be sufficient.

The most closely related result previously known is that a similar nonrecursive relationship exists between the succinctness of descriptions of regular sets by finite automata and (ambiguous) context-free languages respectively (Meyer and Fischer, 1971). In contrast a fairly precise recursive relationship is known to exist between finite automata and deterministic pushdown automata (Stearns, 1967; Meyer and Fischer, 1971; Valiant, 1975). However, the two further questions of analogously relating finite automata with unambiguous grammars, and unambiguous grammars with ambiguous ones, both remain open.

2. Preliminaries

Comparisons between the succinctness of different formalisms can be quantified within the following general framework. Let X be a set on which there is an equivalence relation $\sim$ and a size measure σ mapping X into the positive integers. Then the *maximum succinctness of* X_j *over* X_i for $X_i, X_j \subset X$ can be expressed by the function S_{ij} defined by

$$S_{ij}(n) = \max\{M_i(x) \mid \sigma(x) \leqslant n,\ x \in X_j\},$$

where $M_i(x) = \min\{\sigma(y) \mid y \in X_i,\ y \sim x\}$, and min and max take the value zero for the empty set.

For the case $X_i \subset X_j \subset X_k$ the following general relationships follow immediately from the definitions.

Lemma 1. *For all* n, $S_{ij}(n) \leqslant S_{ik}(n)$.

Lemma 2. *For all* n, $S_{ik}(n) \leqslant S_{ij}(S_{jk}(n))$.

For our applications we want X_1, X_2, X_3, X_4 to correspond to the class of finite automata, deterministic pushdown automata, unambiguous context-free grammars, and general context-free grammars, respectively. The equivalence relation $\sim$ will hold between two elements if and only if they describe the same language. For uniformity we define X_1 to be the class of Chomsky type 3 grammars obtained from finite automata (as in Hopcroft and Ullman, 1969, p. 34) and X_2 to be the union of X_1 and class of canonical grammars obtained from deterministic pushdown automata (e.g., Hopcroft and Ullman, 1969, p. 76). It is well known that for all conventional size measures, both classes of grammars are recursively related to their corresponding automata. We therefore have four classes of grammars $(X_1 \subset X_2 \subset X_3 \subset X_4)$ in which we can define σ uniformly. From among a number of possible size measures for a grammar G (e.g., Gruska, 1972; Ginsburg and Lynch, 1975) we choose $\sigma(G)$ to be the total number of occurrences of terminal and nonterminal symbols in the productions of G.

The following definition of a deterministic pushdown automaton (dpda) is a convenient standard form into which all the other customary formulations can be recursively translated. A dpda M is specified by a sextuple $(Q, \Gamma, \Sigma, \Delta, c_s, F)$, where Q, Γ, Σ are finite sets of *states* $\{s,...\}$, *stack symbols* $\{A,...\}$, and *input symbols* $\{a,...\}$, and Δ, c_s, and F are the transitions, starting configuration, and accepting modes, respectively, as defined below.

Typically we denote words from Γ^* and Σ^* by ω and α, respectively, and their lengths by $|\omega|$ and $|\alpha|$. A *configuration* c is a pair (s, ω) from $Q \times \Gamma^*$, and its height $|c|$ is defined to be $|\omega|$. A *mode*, designated either a *reading mode* or an *ϵ-mode*, is a pair from $Q \times (\Gamma \cup \{\Omega\})$, where Ω is a special empty stack symbol. Δ is a set of transitions, each of the form

$$(s, A) \xrightarrow{\pi} (s', \omega),$$

where $\pi \in \Sigma \cup \{\epsilon\}$ and $|\omega| \leqslant 2$, such that

(i) if (s, A) is a reading mode then for each $a \in \Sigma$ it has a unique transition with $\pi = a$ but none with $\pi = \epsilon$; and

(ii) if (s, A) is an ϵ-mode then it has just one transition, and in this $\pi = \epsilon$.

This machine makes a *move*

$$(s, \omega A) \xrightarrow{\pi} (s', \omega\omega')$$

if and only if there is some transition

$$(s, A) \xrightarrow{\pi} (s', \omega').$$

If $\pi \in \Sigma$ then this symbol is considered to have been read.

A *derivation* is a sequence of such moves through successive configurations and is said to be an *α-derivation* if α is the concatenation of the symbols read by the constituent moves. c is a *stacking configuration* of the derivation if all the configurations following it have height $> |c|$. It is a *popping configuration* of the derivation if all the configurations preceding it have height $> |c|$.

A special configuration of height 1 is designated the starting configuration c_s. The set F of accepting modes is a set of reading modes. A word α is accepted by M if there is an α-derivation from c_s to some c' with mode (i.e., state, top stack symbol) belonging to F.

For a dpda M we denote the cardinalities of Q and Γ by q and t, respectively.

3. Properties of Pushdown Automata

We start with a lemma that illustrates a basic technical argument applicable to deterministic pushdown automata.

LEMMA 3. *Suppose for some configurations* c_0, c_1, *and* c_2 *of* dpda M *such that* $|c_0|$, $|c_2| < m$, $|c_1| > n$, *and* $n - m > q^2t$ *there is* γ*-derivation from* c_0 *to* c_1 *and a* δ*-derivation from* c_1 *to* c_2. *Then for some* β *strictly shorter than* $\gamma\delta$ *there is a* β *derivation from* c_0 *to* c_2.

Proof. Since $n - m > q^2t$ two integers i, j $(m \leqslant i < j \leqslant n)$ clearly exist such that the modes of the stacking configurations in the γ-derivation are the same at heights $i + 1$ and $j + 1$, and the states of the popping configurations of heights i and j in the δ-derivation are also identical. By removing from $\gamma\delta$ the substrings that induce the subderivations between these two pairs of configurations a shorter string β with the desired properties is clearly obtained. ∎

Using the above style of argument in various ways we can prove the following.

LEMMA 4. *There is a positive constant k such that if for some* dpda M *with* q *states and* t *stack symbols* α *is a shortest string such that both* αa *and* αb *are accepted, then* $qt \geqslant (\log |\alpha|)^k$.

Proof. Consider the derivation of M from c_s that reads α and reaches a reading mode c. Let c_m be one of the configurations in this derivation of maximal height. Let c_a, c_b be the configurations with reading modes in F reached when αa, αb, respectively, are read.

If α has the assumed minimality property then clearly no configuration in the α-derivation can repeat. Consequently $|\alpha| \leqslant qt^{|c_m|}$ if $t > 1$ and $|\alpha| \leqslant q \cdot |c_m|$ if $t = 1$. To prove the lemma it remains to show that $|c_m| \leqslant 4q^3t$ which then implies that $qt > (\frac{1}{2}) \cdot (\log |\alpha|)^{1/3}$ for all q, t.

We assume the contrary, that $|c_m| > 4q^3t$ and deduce that in all the following four cases, which are clearly exhaustive, some substrings of α can be removed to give shorter strings still with the required property:

(i) If $|c_m| - |c| > q^2t$ then Lemma 3 (with $c_0 = c_s$, $c_1 = c_m$, $c_2 = c$, $\gamma\delta = \alpha$) can be applied directly to find a β ($|\beta| < |\alpha|$) such that βa and βb are still both accepted.

(ii) If $|c| - \max\{|c_a|, |c_b|\} > q^3t$ (i.e., the readings of both αa and αb are followed by long ϵ-derivations) then extending the argument of Lemma 3 to find a stack segment corresponding to state repetitions in the popping configurations of both ϵ-derivations as well as mode repetitions in the stacking configurations leads to a contradiction similar to (i).

(iii) If $\max\{|c_a|, |c_b|\} - \min\{|c_a|, |c_b|\} > q^2t$ a stack segment exists that corresponds to state repetitions in the popping configurations of the longer ϵ-derivation and mode repetitions in the stacking configurations. This yields the required contradiction.

(iv) If $\min\{|c_a|, |c_b|\} > qt$ then a substring of α corresponding to mode repetitions in the stacking configurations can be removed to give the contradiction. ▌

4. Succinctness Theorem

To prove our main result we use the idea (due to Meyer and Fischer, 1971) of encoding large Turing machine computations in small grammars.

Theorem. *The function S_{23} is not recursively bounded.*

Proof. Let T be a TM with $|T|$ quadruples that on null input halts in Z steps. Let the function $\mathrm{Next}_T(x)$ be defined at x if and only if x is an instantaneous description (ID) of T, and to have value y if y is the ID following x in a computation of T. Let x_0 be the ID for the starting configuration with null tape and let the set $\{\$, a, b\}$ be disjoint from the alphabet describing the ID's.

Let L' be the set of strings of the form

$$\$ x_0 \$ x_1 \$ \cdots \$ x_n \$,$$

where x_n is a halting ID of T, such that for all k $(0 \leqslant k \leqslant (n-1)/2)$ $x_{2k+1}^R = \mathrm{Next}_T(x_{2k})$. Let L'' be the set of strings of the same form under the different restriction that for all k $(1 \leqslant k \leqslant n/2)$ $x_{2k} = \mathrm{Next}_T(x_{2k-1}^R)$. Now let $L = L'a \cup L''b$.

It is easily verified that L', L'' are both recognized by dpda's (say M', M'', respectively) of size polynomial in $|T|$. It follows that $L'a$ and $L''b$ are also both so recognizable and therefore both generated by unambiguous grammars of similar size. Since these languages are disjoint their union L is also generated by an unambiguous grammar of size recursive in $|T|$.

However, L itself is recognized by a dpda. The dpda uses its finite state control to deal with strings of length no more than Z^2 and for longer strings acts as follows. It reads the first Z^2 or so characters of the input α and determines to which of $L'a$ or $L''b$ α may still possibly belong. This initial segment cannot be a prefix of words in both languages for that would imply that a computation of T of more than Z steps on null input exists. If it does not prefix words in either language α is rejected. Otherwise the dpda simulates the rest of α on M' (or M'' as appropriate) and accepts it if after reaching an accepting mode of M' (or M'') an a (or b) follows.

It remains to observe that for any dpda recognizing L, $qt > (\log Z)^k$. This is because the string corresponding to the correct computation sequence of T on null input satisfies the condition of Lemma 4. From the undecidability of the halting problem for TM's we know that Z is not recursively bounded by $| T |$, and hence neither is qt. The family of languages obtained from halting TM's in the manner of L above therefore ensures that S_{23} is not recursively bounded. ▮

Note that since the reversal of L is recognized by a dpda of size recursive in $| T |$, this example also shows that the relative sizes of dpda's required to recognize a language and its reversal are not recursively bounded. As a further application the example can also be used to deduce that it is undecidable whether an unambiguous grammar generates a deterministic language.

5. Summary

We have investigated within a general framework the succinctness relationships among four classes of context-free grammars. Our result is that S_{23} is not recursively bounded. This follows an analogous result for S_{14} by Meyer and Fischer (1971). The recursiveness of S_{12} was first proved by Stearns (1967) and that it is in fact double exponential follows from Meyer and Fischer (1971) and Valiant (1975). That S_{24} is not recursively bounded is immediate from the result for S_{14} together with Lemma 2, or alternatively from that for S_{23} together with Lemma 1. The natures of S_{13} and of S_{34} remain unknown, although the result for S_{14} and Lemma 2 together implies that at least one of them is not recursively bounded.

Acknowledgment

The author is grateful to Albert Meyer and Yoshihide Igarashi for their helpful comments on this paper.

Received: September 25, 1975; revised: January 29, 1976

References

Aho, A. V., and Ullman, J. D. (1972), "The Theory of Parsing, Translation and Compiling," Prentice-Hall, Englewood Cliffs, N. J.

Earley, J. (1970), An efficient context-free parsing algorithm, *Comm. Assoc. Comput. Mach.* 13, 94–104.

Ginsburg, S., and Lynch, N. (1975), Comparative complexity of grammar forms, *in* "7th ACM Symposium on Theory of Computing," Albuquerque, N. M., May 5–7, 1975.

Gruska, J. (1972), On the size of context-free grammars, *Kybernetica* 8, 213–218.

Harrison, M. A., and Havel, I. M. (1973), Strict deterministic grammars, *J. Comput. System Sci.* **7**, 237–277.

Hopcroft, J. E., and Ullman, J. D. (1969), "Formal Languages and Their Relation to Automata," Addison–Wesley, Readmg, Mass.

Knuth, D. E. (1965), On the translation of languages from left to right, *Inform. Contr.* 8, 607–639.

Meyer, A. R., and Fischer, M. J. (1971), Economy of descriptions by automata, grammars and formal systems, *in* "12th Annual Symposium on Switching and Automata Theory," pp. 188–191.

Stearns, R. E. (1967), A regularity test for pushdown machines, *Inform. Contr.* **11**, 323–340.

Valiant, L. G. (1975), Regularity and related problems for deterministic pushdown automata, *J. Assoc. Comput. Mach.* 22, 1–10.