INFORMATION AND CONTROL 69, 255–260 (1986)

A Simplified Lower Bound for Context-Free-Language Recognition*

JOEL I. SEIFERAS

Computer Science Department, University of Rochester, Rochester, New York 14627

For on-line recognition of the words in an arbitrary linear context-free language, there are known tight bounds on the time required by a deterministic multitape Turing machine. In terms of word length n, the time need never be worse than some constant times n^2, even if only one worktape is available; and there is a linear context-free language that requires at least time proportional to $n^2/\log n$, no matter how many worktapes are available. Using Kolmogorov's notion of descriptional complexity as a tool, we present a simple proof of the latter result. © 1986 Academic Press, Inc.

For on-line recognition of the words in an arbitrary linear context-free language (Harrison, 1978), there are known tight bounds on the time required by a deterministic multitape Turing machine. In terms of word length n, the time need never be worse than some constant times n^2, even if only one worktape is available (Kasami, 1967); and there is a linear context-free language that requires at least time proportional to $n^2/\log n$, no matter how many worktapes are available (Gallaire, 1969). Using Kolmogorov's notion of descriptional complexity as a tool, we present a proof of the latter result that is much simpler than the counting argument given by Gallaire (1969).

The linear context-free language L in our proof is essentially the same one Gallaire uses:

$$L = \{ y\$x_1 \not\cdots \not x_k \mid k \geqslant 0;\ x_1, \ldots, x_k, y \in \{0, 1\}^*; \text{ and}$$
$$y = ux_i^R v \text{ for some } i < k \text{ and } u, v \in \{0, 1\}^* \},$$

where w^R denotes the reverse of word w. To recognize a language *on-line*, a Turing machine must indicate before each successive input symbol is received whether the input word so far belongs to the language. For the particular language L, the indications will be "monotonic," changing at most once, from "no" to "yes."

* This work was supported in part by the National Science Foundation, under grant number MCS–8110430.

643/69/1-3-17

0019-9958/86 $3.00

For each 1-dimensional multitape Turing machine M that recognizes L on-line, our new proof specifies a word (in fact, exponentially many words) of each length n that forces M to run for at least some fixed positive fraction times $n^2/\log n$ steps. (Since the choice of the fixed fraction can compensate for any finite number of special cases, we need actually consider only "large" lengths n.) For this purpose, we replace Gallaire's complicated counting argument with the approach suggested by Paul (1979) and by Paul, Seiferas, and Simon (1981), considering individual words in which y is "algorithmically incompressible."

Following Kolmogorov, we define algorithmic incompressibility in terms of descriptional complexity. Any computable partial function $F: \{0, 1\}^* \to \{0, 1\}^*$ can be viewed as a description scheme, in terms of which we can define a descriptional complexity $K_F: \{0, 1\}^* \to \{0, 1, 2, ..., \infty\}$ by

$$K_F(x) = \min\{|d| \mid F(d) = x\}.$$

(Note that we do not require our descriptions to be self-delimiting or prefix free: $F(d)$ and $F(de)$ might both be defined, but yet be different.) Because there is a "universal" computable partial function, there is some F_0 for which

$$\forall F \exists c_F \forall x K_{F_0}(x) \leqslant K_F(x) + c_F.$$

Except for an additive constant, therefore, F_0 is as succinct a description scheme as any; so we define *the* descriptional complexity $K(x)$ of x to be $K_{F_0}(x)$. A word x is *(algorithmically) incompressible* if $K(x) \geqslant |x|$. Since there are 2^n binary words of length n but only $2^n - 1$ possible shorter descriptions d, there is sure to be at least one incompressible word of each length. In fact, if we relax our standards of incompressibility by even 1 ($K(x) \geqslant |x| - 1$), which is still quite sufficient for our purposes, then *most* words of each length must qualify as incompressible.

Now let us return to the specification of an input word of length n that is hard for M. Regardless of M, we choose y to be an incompressible word of length $\lfloor n/2 \rfloor$, still leaving half the desired length for prospective reverse subwords x_i. It will suffice, then, to show that we can choose each successive $x_i ¢$ to be of length proportional to only the *logarithm* of n and yet to require *linearly* many additional steps by M. Due to monotonicity, this will require also that each successive x_i *not* be a reverse subword of y.

Assume inductively that $x_1, ..., x_{i-1}$ have been chosen as required, so that the input prefix $y\$x_1 ¢ \cdots ¢ x_{i-1} ¢$ does not yet belong to L. As a consequence of Lemmas 1 and 2 below, for some appropriate constant c, y is the only word of its length $\lfloor n/2 \rfloor$ with precisely its set of subwords of length $m = 2 \log_2 n + c$. It follows from this that the $¢$-termination $x_i ¢$ of

some length-m word x_i that is not a reverse subword of y must require at least $t = \varepsilon n$ many additional steps by M, where ε is a positive fraction that does not depend on n. Otherwise, we could devise a short description of the length-m subwords of y, and hence of y itself, based primarily on a relatively small portion of the instantaneous description of M when it is about to read x_i; it could be determined separately whether each length-m word is a subword of y by continuing the computation by M for t steps from the provided partial instantaneous description, with the $\not{c}$-terminated reverse of that candidate word as input continuation—by assumption, any such continuation for more than t steps could safely be cut short, serving already as decisive indication that the candidate *is* a subword of y. In addition to the worktape contents within distance t of the tape heads, we would have to include only the following: this whole discussion (suitably formalized), specification of M, the length n, the instantaneous control state of M, and the instantaneous locations of M's tape heads on the provided worktape fragments. If ε is small in terms of M, and if n is large, then all this does add up to fewer than $\lfloor n/2 \rfloor$ bits, as it should not. Except for Lemmas 1 and 2, this concludes our proof by contradiction that each successive x_i can be chosen to be sufficiently time-consuming to yield the desired lower bound.

Finally, we turn to Lemmas 1 and 2. A *repetition* is a subword that occurs in two distinct, but possibly overlapping, positions; i.e., x is a repetition in w if $u_1 x v_1 = w = u_2 x v_2$ for distinct prefixes u_1 and u_2.

LEMMA 1. *If a word has no repetition of length m, then it is determined by its subwords of length* $m+1$; *i.e., then it is the unique word with no repetition of length m and with precisely its set of subwords of length* $m+1$.

Proof. If x is a word and a and b are single characters, then call ax a *left neighbor* of xb, and call xb a *right neighbor* of ax. If a word has no repetition of length m, then every subword of length $m+1$ has at most one right neighboring subword, so that the next letter (if any) following its unique occurrence is determined. The word's *prefix* of length $m+1$ is the unique subword of that length with no left neighboring subword. By induction, therefore, the entire word is determined by its subwords of length $m+1$. ∎

LEMMA 2. *An incompressible word of length n has no repetition longer than* $2 \log_2 n$ *plus some constant c.*

Proof. If a repetition's length is a slightly larger multiple of $\log_2 n$, then we can easily obtain a too-short description of the entire word by replacing one instance with a clear enough reference to the other. Although such a

lemma would be strong enough for our purposes, it is interesting to ask just how long a repetition is possible and to prove the best result we can.

Let $w = xyz$ be the entire incompressible word of length n, and suppose y is a repetition of length $2\lceil \log_2 n \rceil + c$ in xy. To describe w, we concatenate the following information: this discussion (suitably formalized), the value of c, the location (less than $|x|$) of the beginning of the first instance of y in xy, the location ($|x|$) of the beginning of the last instance of y in xy, and the shortened word xz. The first item should be represented by a self-delimiting word, say c' bits long. The second item should also be self-delimiting, at most $3 \log_2 c$ bits long, say. The two locations should be binary radix representations, each padded with insignificant high-order zeros to length exactly $1 + \lceil \log_2 n \rceil$. This way, the total length of the three items following the two self-delimiting ones will be exactly $n - c + 2$, so that n can be inferred and the entire description correctly parsed. The description is too short (and the proof by contradiction complete) provided $c - 2 > c' + 3 \log_2 c$. ∎

It is interesting to note, on the other hand, that every incompressible word of length n does have a repetition of length about $\log_2 n$. In fact *every* binary word of large enough length n has a repetition of length $m = \lfloor \log_2 n \rfloor - 1$, since there are only $2^m \leqslant n/2$ distinct subwords of that length possible. For an incompressible word, we can increase the guaranteed repetition length by at least 1:

PROPOSITION. *Every incompressible word of large enough length n has a repetition of length at least* $\lfloor \log_2 n \rfloor$.

Proof. Suppose, to the contrary, that $w = xy$ is an incompressible word of length n with no repetition of length $m = \lfloor \log_2 n \rfloor$, and suppose y is its suffix of length m. Since $n - 2m$ subwords of length m already occur in x, there are at most $2^m - n + 2m \leqslant 2m$ possibilities for y, given x. If n is large enough, therefore, the following suffice to describe w too succinctly: this discussion (suitably formalized, self-delimiting), the value of m (binary radix, self-delimiting), the serial number of y among those words of length m that are not subwords of the prefix x (binary radix, self-delimiting), and the literal word x. The savings by omitting y is proportional to $\log n$, while the nonliteral replacement is proportional to only some constant plus $\log \log n$. ∎

For y, Gallaire always uses a *de Bruijn sequence*, a word of some length 2^m in which each word of length m occurs contiguously (counting "wrap-around") exactly once. It follows from the proposition, therefore, that the words we use are in fact different from the ones Gallaire uses. Lemma 1 shows that the de Bruijn criterion is unnecessarily stringent; the crucial

requirement is only that no subword of the appropriate length (any constant time $\log n$ gives the same quantitative result) occurs more than once, not that every one does occur.

For the counting argument he presents (only partly, actually, citing Hennie, 1966, for elaboration), Gallaire must cite clever arguments that enough de Bruijn words exist (de Bruijn, 1946; Golomb, 1967; Hall, 1967). (For a more recent survey, see Fredricksen, 1982.) While incompressible words are much more obviously abundant, on the other hand, we no longer even need that abundance, since the powerful incompressibility assumption enables us to focus exclusively on one particular word y. The result is the clear and self-contained proof presented above.

Like Gallaire's, unfortunately, our lower-bound argument does depend on both the on-line restriction and the limited architecture of the multitape Turing machine. A *random-access* machine can recognize L in *linear* time, by building the tree of all subwords of the reverse of y in time linear in $|y|$ (Weiner, 1973; McCreight, 1976; Chen and Seiferas, 1985) and then searching down through it for each successive x_i in time linear in $|x_i|$. *Off-line*, a multitape Turing machine can recognize the very similar linear context-free language

$$L' = \{ y\$x_1 \not\$ \cdots \not\$ x_k \mid k \geq 0;\ x_1, \ldots, x_k, y \in \{0, 1\}^*; \text{ and } y = u x_i^R v \text{ for } i = k - 1 \text{ and some } u, v \in \{0, 1\}^* \}$$

in linear time, using the Fischer–Paterson implementation (1974) of the Knuth–Morris–Pratt string-matching algorithm (1977). Note that the lower-bound argument does still apply to L'. (For L', in fact, the argument is slightly easier, since each x_i in the tail of the hard input word need not fail to be a reverse subword of y. On-line, L sounds like *it* might be harder; but, off-line, L' sounds like *it* might be harder.)

On Turing machines with *multidimensional* "tapes," our argument still does yield nontrivial lower bounds, but they are not as close to any known upper bounds. In the argument, if M has d-dimensional tapes, then we can obtain the desired too-short description of y if we assume the time t needed for each next x_i is bounded by a small enough fraction of $n^{1/d}$. This yields a lower bound proportional to $n^{1+1/d}/\log n$ and raises the question of whether the Kasami upper bound (1967) can be improved using multidimensional tapes.

ACKNOWLEDGMENTS

Michael Harrison first brought Gallaire's argument to my attention. M.T. Chen helped me to appreciate the role of de Bruijn's theorem in that argument and pointed out that the

argument must fail for a random-access machine. Larry Ruzzo pointed out that the variant L' would work as well as Gallaire's language. Laura Sanchis and Zvi Galil provided constructive criticism of previous versions of the manuscript.

RECEIVED January 29, 1986

REFERENCES

DE BRUIJN, N. G. (1946), A combinatorial problem, *Nederl. Akad. Wetensch. Proc.* **49**, 758–764.

CHEN, M. T., AND SEIFERAS, J. I. (1985), Efficient and elegant subword-tree construction, *in* "Combinatorial Algorithms on Words" (A. Apostolico and Z. Galil, Eds.), pp. 97–107, Springer-Verlag, New York/Berlin.

FISCHER, M. J., AND PATERSON, M. S. (1974), String-matching and other products, *in* "Complexity of Computation" (R. M. Karp, Ed.), pp. 113–125, Amer. Math. Soc., Providence, R.I.

FREDRICKSEN, H. (1982), A survey of full length nonlinear shift register cycle algorithms, *SIAM Rev.* **24**, 195–221.

GALLAIRE, H. (1969), Recognition time of context-free languages by on-line Turing machines, *Inform. Contr.* **15**, 288–295.

GOLOMB, S. W. (1967), "Shift Register Sequences," p. 131 ff, Holden–Day, San Francisco.

HALL, M., JR. (1967), "Combinatorial Theory," pp. 91–99, Ginn (Blaisdell), Boston.

HARRISON, M. A. (1978), "Introduction to Formal Language Theory," Chap. 12, Addison–Wesley, Reading, Mass.

HENNIE, F. C. (1966), On-line Turing machine computations, *IRE Trans. Electron. Comput.* **EC-15**, 35–44.

KASAMI, T. (1967), A note on computing time for recognition of languages generated by linear grammars, *Inform. Contr.* **10**, 209–214.

KNUTH, D. E. MORRIS, J. H., JR., AND PRATT, V. R. (1977), Fast pattern matching in strings, *SIAM J. Comput.* **6**, 323–350.

MCCREIGHT, E. M. (1976), A space-economical suffix tree construction algorithm, *J. Assoc. Comput. Mach.* **23**, 262–272.

PAUL, W. J. (1979), Kolmogorov complexity and lower bounds, *in* "Second International Conference on Fundamentals of Computation Theory" (L. Budach, Ed.), pp. 325–334, Akademie-Verlag, Berlin.

PAUL, W. J., SEIFERAS, J. I., AND SIMON, J. (1981), An information-theoretic approach to time bounds for on-line computation, *J. Comput. System Sci.* **23**, 108–126.

WEINER, P. (1973), Linear pattern matching algorithms, *in* "14th Annual Symposium on Switching & Automata Theory," IEEE Computer Society, Long Beach, California, pp. 1–11.