SIAM J. COMPUT.
Vol. 24, No. 2, pp. 398–410, April 1995

A NEW APPROACH TO FORMAL LANGUAGE THEORY BY KOLMOGOROV COMPLEXITY*

MING LI† AND PAUL VITÁNYI‡

Abstract. We present a new approach to formal language theory by using Kolmogorov complexity. The main results presented here are an alternative for pumping lemma(s), a new characterization for regular languages, and a new method to separate deterministic context-free languages and nondeterministic context-free languages. The use of the "incompressibility arguments" is illustrated by many examples. The approach is also successful at the high end of the Chomsky hierarchy since one can quantify nonrecursiveness in terms of Kolmogorov complexity.

Key words. formal language theory, Kolmogorov complexity, pumping lemmas, regular languages, finite automata, deterministic context-free languages

AMS subject classifications. 68Q30, 68Q45, 68Q68, 68Q50

1. Introduction.

It is feasible to reconstruct parts of formal language theory by using algorithmic information theory (Kolmogorov complexity). We provide theorems on how to use Kolmogorov complexity as a concrete and powerful tool. We do not just want to introduce fancy mathematics; our goal is to help our readers do a large part of formal language theory in the most essential, easiest, and sometimes even obvious ways. In this paper, it is only important to us to demonstrate that the application of Kolmogorov complexity in the targeted area is not restricted to trivialities. The proofs of the theorems in this paper may not be easy. However, the theorems are the type that are used as a tool. Once derived, our theorems are easy to apply.

1.1. Prelude.

The first application of Kolmogorov complexity in the theory of computation was in [18] and [19]. By redoing proofs of known results, it was shown that static, descriptional (program size) complexity of a *single* random string can be used to obtain lower bounds on dynamic, computational (running time) complexity. None of the inventors of Kolmogorov complexity originally had these applications in mind. Recently, Kolmogorov complexity has been applied extensively to solve classic open problems of sometimes two decades standing [15], [11], [8], [9]. For more examples, see the textbook [12].

The secret of Kolmogorov complexity's success in dynamic, computational lower bound proofs rests on a simple fact: the overwhelming majority of strings has hardly any computable regularities. We call such a string "Kolmogorov random" or "incompressible." A Kolmogorov random string cannot be (effectively) compressed. Incompressibility is a noneffective property: no individual string, except finitely many, can be proved incompressible.

Recall that a traditional lower bound proof by counting usually involves *all* inputs of certain length. One shows that a certain lower bound has to hold for *some* "*typical*" input. Since an individual typical input is *hard* (sometimes impossible) to find, the proof must involve all the inputs. Now we understand that a typical input of each length can be constructed via

*Received by the editors February 22, 1993; accepted for publication November 12, 1993. A preliminary version of part of this work was presented at the 16th *International Colloquium on Automata, Languages, and Programming*, Stresa, Italy, July 1989.

†Current address, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1 (mli@math.waterloo.edu). The research of this author was supported in part by National Science Foundation grant DCR-8606366, Office of Naval Research grant N00014-85-k-0445, Army Research Office grant DAAL03-86-K-0171, and by NSERC operating grants OGP-0036747 and OGP-046506. Part of this work was performed while the author was at the Department of Computer Science, York University, North York, Ontario, Canada.

‡CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands (paulv@cwi.nl). Partially supported by NSERC International Scientific Exchange Award ISE0046203, and by NWO through NFI Project ALADDIN under contract NF 62-376.

an incompressible string. However, only finitely many individual strings can be effectively proved to be incompressible. No wonder the old counting arguments had to involve all inputs. In a proof using the "incompressibility method," one uses an individual incompressible string that is known to *exist* even though it cannot be constructed. Then one shows that if the assumed lower time bound would not hold, then this string could be compressed, and hence it would not be incompressible.

1.2. Outline of the paper. The incompressibility argument also works for formal languages and automata theory. Assume the basic notions treated in a textbook such as [6].

The first result is a powerful alternative to pumping lemmas for regular languages. It is well known that not all nonregular languages can be shown to be nonregular by the usual uvw-pumping lemma. There is a plethora of pumping lemmas to show nonregularity, like the "marked pumping lemma," and so on. In fact, it seems that many example nonregular languages require their own special purpose pumping lemmas. Recently, [7], [21], [3], exhaustive pumping lemmas that characterize the regular languages have been obtained.

These pumping lemmas are complicated and complicated to use. The last reference uses Ramsey theory. In contrast, by using Kolmogorov complexity we give a new characterization of the regular languages that simply makes our intuition of the "finite stateness" of these languages rigorous and easy to apply. Since it is a characterization, it works for all nonregular languages. We give several examples of its application, some of which were quite difficult using pumping lemmas.

To prove that a certain context-free language (cfl) is not deterministic context-free (dcfl) has required laborious ad-hoc proofs [6], or cumbersome and difficult pumping lemmas or iteration theorems [4], [24]. We give necessary (Kolmogorov complexity) conditions for dcfl, that are very easy to apply. We test the new method on several examples in cfl–dcfl, which were hard to handle before. In certain respects the KC-DCFL lemma may be more powerful than the related lemmas and theorems mentioned above. On the high end of the Chomsky hierarchy we present, for completeness, a known characterization of recursive languages, and a necessary condition for recursively enumerable languages.

2. Kolmogorov complexity. From now on, let x denote both the natural number and the xth binary string in the sequence 0, 1, 00, 01, 10, 11, 000, ... That is, the representation "3" corresponds both to the natural number 3 and to the binary string 00. This way we obtain a natural bijection between the nonnegative integers $\mathcal{N}$ and the finite binary strings $\{0, 1\}^*$. Numerically, the binary string $x_{n-1} \ldots x_0$ corresponds to the integer

$$2^n - 1 + \sum_{i=0}^{n-1} x_i 2^i. \tag{1}$$

We use notation $l(x)$ to denote the *length* (number of bits) of a binary string x. If x is not a finite binary string but another finite object like a finite automaton, a recursive function, or a natural number, then we use $l(x)$ to denote the length of its standard binary description. Let $\langle \cdot, \cdot \rangle : \mathcal{N} \times \mathcal{N} \rightarrow \mathcal{N}$ be a standard recursive, invertible, one-one encoding of pairs of natural numbers in natural numbers. This idea can be iterated to obtain a pairing from triples of natural numbers with natural numbers $\langle x, y, z \rangle = \langle x, \langle y, z \rangle \rangle$, and so on.

Any of the usual definitions of Kolmogorov complexity in [10], [19], and [12] will do for the sequel. We are interested in the shortest effective description of a finite object x. To fix thoughts, consider the problem of describing a string x over 0's and 1's. Let $T_1, T_2, \ldots$ be the standard enumeration of Turing machines. Since T_i computes a partial recursive function $\phi_i : \mathcal{N} \rightarrow \mathcal{N}$, we obtain the standard enumeration $\phi_1, \phi_2, \ldots$ of partial recursive functions.

We denote $\phi(\langle x, y\rangle)$ as $\phi(x, y)$. Any partial recursive function ϕ from strings over 0's and 1's to such strings, together with a string p, the *program* for ϕ to compute x, such that $\phi(p) = x$, is a description of x. It is useful to generalize this idea to the conditional version: $\phi(p, y) = x$ such that p is a program for ϕ to compute x, given a binary string y for free. Then the *descriptional* complexity C_ϕ of x, *relative* to ϕ and y, is defined by

$$C_\phi(x|y) = \min\{l(p) : p \in \{0,1\}^*, \phi(p, y) = x\},$$

or ∞ if no such p exists.

For a *universal* partial recursive function ϕ_0, computed by the universal Turing machine U, we know that, for each partial recursive function $\phi = Q_i$ there is a constant c_ϕ such that for all strings x, y, we have $\phi_0(i, x, y) = \phi(x, y)$. Hence, $C_{\phi_0}(x|y) \leq C_\phi(x|y) + c_\phi$. We fix a reference universal function ϕ_0 and define the *conditional Kolmogorov complexity* of x given y as $C(x|y) = C_{\phi_0}(x|y)$.[1]

The *unconditional Kolmogorov complexity* of x is $C(x) = C(x|\epsilon)$, where ϵ denotes the empty string ($l(\epsilon) = 0$).

Since there is a Turing machine that just copies its input to its output we have $C(x|y) \leq l(x) + O(1)$, for each x and y. Since there are 2^n binary strings of length n, but only $2^n - 1$ possible shorter descriptions, it follows that $C(x) \geq l(x)$ for some binary string x of each length. We call such strings *incompressible* or *random* [16], [17]. It also follows that for each length n and each binary string y, there is a binary string x of length n such that $C(x|y) \geq l(x)$. Considering C as an integer function, using the obvious one-one correspondence between finite binary words and nonnegative integers, it can be shown that $C(x) \to \infty$ for $x \to \infty$. Finally, $C(x, y)$ denotes $C(\langle x, y\rangle)$.

EXAMPLE 1 (self-delimiting strings). A *prefix code* is a mapping from finite binary code words to source words such that no code word is a proper prefix of any other code word. We define a particular prefix code.

For each binary source word $x = x_1 \ldots x_n$, define the code word $\bar{x}$ by

$$\bar{x} = 1^{l(x)}0x.$$

Define

$$x' = \overline{l(x)}x.$$

The string x' is called the *self-delimiting* code of x.

Set $x = 01011$. Then $l(x) = 5$, which corresponds to binary string "10," and $\overline{l(x)} = 11010$. Therefore, $x' = 1101001011$ is the self-delimiting code of "01011."

The self-delimiting code of a positive integer x requires $l(x) + 2\log l(x) + 1$ bits. It is easy to verify that $l(x) = \lfloor \log(x+1) \rfloor$. All logarithms are base 2 unless otherwise noted. For convenience, we simply replace the length $l(x)$ of a natural number x by "$\log x$."

EXAMPLE 2 (substrings of incompressible strings). Is a substring of an incompressible string also incompressible? A string $x = uvw$ can be specified by a short description for v of length $C(v)$, a description of $l(u)$, and the literal description of uw. Moreover, we need information to tell these three items apart. Such information can be provided by prefixing each item with a self-delimiting description of its length. Together this takes $C(v) + l(uw) + O(\log l(x))$ bits. Hence,

$$C(x) \leq C(v) + O(\log l(x)) + l(uw).$$

[1] Similarly, we define the complexity of the xth partial recursive function ϕ conditional to the yth partial recursive function ψ by $C(\phi|\psi) = C(x|y)$.

Thus, if we choose x incompressible, $C(x) \geq l(x)$, then we obtain

$$C(v) \geq l(v) - O(\log l(x)).$$

It can be shown that this is optimal—some substring of an incompressible string of length n may be compressible by an $\Omega(\log n)$ additional term. This conforms to a fact we know from probability theory: every random string of length n is expected to contain a run of about $\log n$ consecutive zeros (or ones). Such a substring has complexity $O(\log\log n)$.

3. Regular sets and finite automata.

DEFINITION 3.1. *Let Σ be a finite nonempty alphabet, and let Q be a (possibly infinite) nonempty set of states. A* transition function *is a function $\delta : \Sigma \times Q \to Q$. We extend δ to δ' on Σ^* by $\delta'(\epsilon, q) = q$ and*

$$\delta'(a_1 \dots a_n, q) = \delta(a_n, \delta'(a_1 \dots a_{n-1}, q)).$$

Clearly, if δ' is not $1-1$, then the automaton "forgets" because some x and y from Σ^* drive δ' into the same memory state. An automaton *A is a quintuple $(\Sigma, Q, \delta, q_0, q_f)$, where everything is as above and $q_0, q_f \in Q$ are distinguished as* initial state *and* final state*, respectively. We call A a* finite automaton *(fa) if Q is finite.*

We denote "indistinguishability" of a pair of histories $x, y \in \Sigma^*$ by $x \sim y$, defined by $\delta'(x, q_0) = \delta'(y, q_0)$. "Indistinguishability" of strings is reflexive, symmetric, transitive, and right-invariant ($\delta'(xz, q_0) = \delta'(yz, q_0)$ for all z). Thus, "indistinguishability" is a right-invariant equivalence relation on Σ^*. It is a simple matter to ascertain this formally.

DEFINITION 3.2. *The language accepted by automaton A as above is the set $L = \{x : \delta'(x, q_0) = q_f\}$. A* regular language *is a language accepted by a finite automaton.*

It is a straightforward exercise to verify from the definitions the following fact (which will be used later).

THEOREM 3.3 (Myhill and Nerode). *The following statements about $L \subseteq \Sigma^*$ are equivalent.*

(i) *$L \subseteq \Sigma^*$ is accepted by some finite automaton.*

(ii) *L is the union of equivalence classes of a right-invariant equivalence relation of finite index on Σ^*.*

(iii) *For all $x, y \in \Sigma^*$ define right-invariant equivalence $x \sim y$ by the following item: for all $z \in \Sigma^*$ we have $xz \in L$ iff $yz \in L$. Then the number of $\sim$-equivalence classes is finite.*

Subsequently, closure of finite automaton languages under complement, union, and intersection follow by simple construction of the appropriate δ functions from given ones. Details can be found in any textbook on the subject such as [6]. The clumsy pumping lemma approach can now be replaced by the Kolmogorov formulation below.

3.1. Kolmogorov complexity replacement for the pumping lemma.

An important part of formal language theory is deriving a hierarchy of language families. The main division is the Chomsky hierarchy, with regular languages, context-free languages, context-sensitive languages and recursively enumerable languages. The common way to prove that certain languages are not regular is by using "pumping" lemmas, for instance, the uvw lemma. However, these lemmas are quite difficult to state and cumbersome to prove or use. In contrast, we show below how to replace such arguments by simple and intuitive, yet rigorous, Kolmogorov complexity arguments.

Regular languages coincide with the languages accepted by finite automata. This invites a straightforward application of Kolmogorov complexity. Let us give an example. We prove that $\{0^k 1^k : k \geq 1\}$ is not regular. If it were, then the state q of a particular accepting fa A after processing 0^k, together with A, is, up to a constant, a description of k. Namely, by running A

initialized in state q on input consisting of only 1's, the first time A enters an accepting state is after precisely k consecutive 1's. The size of the description of A and q is bounded by a constant, say c, which is independent of k. Altogether, it follows that $C(k) \leq c + O(1)$. But choosing k with $C(k) \geq \log k$ we obtain a contradiction for all large enough k. Hence, since A has a fixed finite number of states, there is a fixed finite number that bounds the Kolmogorov complexity of each natural number: contradiction. We generalize this observation as follows.

DEFINITION 3.4. *Let Σ be a finite nonempty alphabet and $\phi : \mathcal{N} \to \Sigma^*$ be a total recursive function. Then ϕ enumerates (possibly a proper subset of) Σ^* in order $\phi(1), \phi(2), \ldots$ We call such an order* effective *and ϕ an* enumerator. *The* lexicographical order *is the effective order such that all words in Σ^* are ordered first according to length, and then lexicographically within the group of each length. Another example is ϕ such that $\phi(i) = p_i$, the standard binary representation of the ith prime, is an effective order in* $\{0, 1\}^*$. *In this case ϕ does not enumerate all of Σ^*. Let $L \subseteq \Sigma^*$. Define $L_x = \{y : xy \in L\}$.*

LEMMA 3.5 (KC regularity). *Let $L \subseteq \Sigma^*$ be regular and let ϕ be an enumerator in Σ^*. Then there exists a constant c that depends only on L and ϕ such that for each x, if y is the nth string enumerated in* (*or in the complement of*) *L_x, then $C(y) \leq C(n) + c$.*

Proof. Let L be a regular language. The nth string y such that $xy \in L$ for some x can be described by

- this discussion and a description of the fa that accepts L,
- a description of ϕ, and
- the state of the fa after processing x, and the number n.

The statement "or in the complement of" follows, since regular languages are closed under complementation. □

As an application of the KC Regularity lemma we prove that $\{1^p : p \text{ is prime}\}$ is not regular. Consider the string $xy = 1^p$ with p the $(k+1)$th prime. Set $x = 1^{p'}$, with p' the kth prime. Then $y = 1^{p-p'}$, and y is the lexicographical first element in L_x. Hence, by Lemma 3.5, $C(p - p') = O(1)$. But the difference between two consecutive primes grows unbounded. Since there are only $O(1)$ descriptions of length $O(1)$, we have a contradiction. We give some more examples from the well-known textbook of Hopcroft and Ullman [6].

EXAMPLE 3 (Exercise 3.1(h)* in [6]). Show that $L = \{xx^R w : x, w \in \{0, 1\}^* - \{\epsilon\}\}$ is not regular. Set $x = (01)^m$, where $C(m) \geq \log m$. Then, the lexicographically first word in L_x is y with $y = (10)^m 0$. But $C(y) = \Omega(\log m)$, which contradicts the KC Regularity lemma.

EXAMPLE 4. Prove that $L = \{0^i 1^j : i \neq j\}$ is not regular. Set $x = 0^m$, where $C(m) \geq \log m$. Then the lexicographically first word *not* in $L_x \bigcap \{1\}^*$ is $y = 1^m$. But $C(y) = \Omega(\log m)$, which contradicts the KC Regularity lemma.

EXAMPLE 5 (Exercise 3.6* in [6]). Prove that $L = \{0^i 1^j : \gcd(i, j) = 1\}$ is not regular. Set $x = 0^{(p-1)!}1$, where $p > 3$ is a prime, $l(p) = n$, and $C(p) \geq n - \log n$. Then the lexicographically first word in L_x is 1^{p-1}, which contradicts the KC Regularity lemma.

EXAMPLE 6 (§2.2, Exercises 11–15 in [4]). Prove that $\{p : p$ is the standard binary representation of a prime $\}$ is not regular. Suppose the contrary, and p_i denotes the ith prime, $i \geq 1$. Consider the least binary $p_m = uv$ $(= u2^{l(v)} + v)$, with $u = \Pi_{i<k} p_i$ and v not in $\{0\}^*\{1\}$. Such a prime p_m exists since each interval $[n, n + n^{11/20}]$ of the natural numbers contains a prime [5].

Consider p_m now as an integer, $p_m = 2^{l(v)} \Pi_{i<k} p_i + v$. Since integer $v > 1$ and v is not divided by any prime less than p_k (because p_m is prime), the binary length $l(v) \geq l(p_k)$. Because p_k goes to infinity with k, the value $C(v) \geq C(l(v))$ also goes to infinity with k. But since v is the lexicographical first suffix, with integer $v > 1$ such that $uv \in L$, we have $C(v) = O(1)$ by the KC Regularity lemma, which is a contradiction.

3.2. Kolmogorov complexity characterization of regular languages. While the pumping lemmas are not precise enough (except for the difficult construction in [3]) to characterize the regular languages, this is easy with Kolmogorov complexity. In fact, the KC Regularity lemma is a direct corollary of the characterization below. The theorem is not only a device to show that some nonregular languages are nonregular, as are the common pumping lemmas, but it is a *characterization* of the regular sets. Consequently, it determines whether or not a given language is regular, just like the Myhill–Nerode theorem. The usual characterizations of regular languages seem to be practically useful only to prove regularity. The need for pumping lemmas stems from the fact that characterizations tend to be very hard to use in showing nonregularity. In contrast, the KC characterization is practicable for both purposes, as shown in the examples.

DEFINITION 3.6. *Let Σ be a nonempty finite alphabet, and let y_i be the ith element of Σ^* in lexicographic order, $i \geq 1$. For $L \subseteq \Sigma^*$ and $x \in \Sigma^*$, let $\chi = \chi_1\chi_2\ldots$ be the* characteristic sequence *of $L_x = \{y : xy \in L\}$, defined by $\chi_i = 1$* if $xy_i \in L$, and $\chi_i = 0$ otherwise. We denote $\chi_1 \ldots \chi_n$ by $\chi_{1:n}$.

THEOREM 3.7 (Regular KC characterization). *Let $L \subseteq \Sigma^*$, and assume the notation above. The following statements are equivalent.*

(i) *L is regular.*

(ii) *There is a constant c_L that depends only on L, such that for all $x \in \Sigma^*$ and for all n, $C(\chi_{1:n}|n) \leq c_L$.*

(iii) *There is a constant c_L that depends only on L, such that for all $x \in \Sigma^*$ and for all n, $C(\chi_{1:n}) \leq C(n) + c_L$.*

(iv) *There is a constant c_L that depends only on L, such that for all $x \in \Sigma^*$ and for all n, $C(\chi_{1:n}) \leq \log n + c_L$.*

Proof. (i) → (ii): By similar proof as the KC Regularity lemma.

(ii) → (iii): obvious.

(iii) → (iv): obvious.

(iv) → (i): shown in the following claim.

CLAIM 3.8. For each constant c there are only finitely many one-way infinite binary strings ω such that for all n, $C(\omega_{1:n}) \leq \log n + c$.

Proof. The claim is a weaker version of Theorem 6 in [2]. It turns out that the weaker version admits a simpler proof. To make the treatment self-contained, we present this new proof in the Appendix. □

By (iv) and the claim, there are only finitely many distinct χ's associated with the x's in Σ^*. Define the right-invariant equivalence relation $\sim$ by $x \sim x'$ if $\chi = \chi'$. This relation induces a partition of Σ^* in equivalence classes $[x] = \{y : y \sim x\}$. Since there is a one-one correspondence between the $[x]$'s and the χ's, and there are only finitely many distinct χ's, there are also only finitely many $[x]$'s, which implies that L is regular by the Myhill–Nerode theorem. □

REMARK 1. The KC Regularity lemma may be viewed as a corollary of the theorem. If L is regular, then clearly L_x is regular, and it follows immediately that there are only finitely many associated χ's, and each can be specified in at most c bits, where c is a constant depending only on L (and enumerator ϕ). If y is, say, the nth string in L_x, then we can specify y as the string corresponding to the nth '1' in χ, using only $C(n) + O(1)$ bits to specify y. Hence $C(y) \leq C(n) + O(1)$. Without loss of generality, we need to assume that the nth string enumerated in L_x in the KC-regularity Lemma is the string corresponding to the nth '1' in χ by the enumeration in the Theorem, or that there is a recursive mapping between the two.

REMARK 2. If L is nonregular, then there are infinitely many $x \in \Sigma^*$ with distinct equivalence classes $[x]$, each of which has its own distinct associated characteristic sequence χ. It is easy to see, for each automaton (finite or infinite) and for each χ associated with an equivalence class $[x]$, we have

$$C(\chi_{1:n}|n) \to \inf\{C(y) : y \in [x]\} + O(1),$$

for $n \to \infty$. The difference between finite and infinite automata is precisely expressed in the fact that only in the first case does there exist an a priori constant which bounds the left-hand term for all χ.

We show how to prove positive results with the KC characterization theorem. (Examples of negative results were given in the preceding section.)

EXAMPLE 7. Prove that $L = \Sigma^*$ is regular. There exists a constant c such that for each x the associated characteristic sequence is $\chi = 1, 1, \ldots$, with $C(\chi_{1:n}|n) \leq c$. Therefore, L is regular by the KC characterization theorem.

EXAMPLE 8. Prove that $L = \{x : x \text{ is accepted by a 2-way dfa}\}$ is regular. There exists a constant c such that for each x we have $C(\chi_{1:n}|n) \leq c$. Therefore, L is regular by the KC Characterization theorem.

4. Deterministic context-free languages. We present a Kolmogorov complexity based criterion to show that certain languages are not dcfl. In particular, it can be used to demonstrate the existence of witness languages in the difference of the family of context-free languages (cfls) and deterministic context-free languages (dcfls). Languages in this difference are the most difficult to identify; other non-dcfl are also non-cfl and in those cases we can often use the pumping lemma for context-free languages. The new method compares favorably with other known related techniques (mentioned in the Introduction) by being simpler, easier to apply, and apparently more powerful (because it works on a superset of examples). Yet our primary goal is to demonstrate the usefulness of Kolmogorov complexity in this matter.

A language is a dcfl iff it is accepted by a deterministic pushdown automaton (dpda).

Intuitively, Lemma 4.2 tries to capture the following. Suppose a dpda accepts $L = \{0^n 1^n 2^n : n \geq 1\}$. Then the dpda needs to first store a representation of the all 0 part, and then retrieve it to check against the all 1 part. But after that check, it seems inevitable that it has discarded the relevant information about n, and cannot use this information again to check against the all 2 part. That is, the complexity of the all 2 part should be $C(n) = O(1)$, which yields a contradiction for large n.

DEFINITION 4.1. *A one-way infinite string* $\omega = \omega_1\omega_2\ldots$ *over* Σ *is* recursive *if there is a total recursive function* $f : \mathcal{N} \to \Sigma$ *such that* $\omega_i = f(i)$ *for all* $i \geq 1$.

LEMMA 4.2 (KC-DCFL). *Let* $L \subseteq \Sigma^*$ *be recognized by a deterministic pushdown machine* M *and let* c *be a constant. Let* $\omega = \omega_1\omega_2\ldots$ *be a recursive sequence over* Σ *which can be described in* c *bits. Let* $x, y \in \Sigma^*$ *with* $C(x, y) < c$ *and let* $\zeta = \ldots\zeta_2\zeta_1$ *be a* (*reversed*) *recursive sequence over* Σ *of the form* $\ldots yyx$. *Let* $n, m \in \mathcal{N}$ *and* $w \in \Sigma^*$ *be such that items* (i)–(iii) *below are satisfied.*

(i) *For each* i $(1 \leq i \leq n)$, *given* M*'s state and pushdown store contents after processing input* $\zeta_m \ldots \zeta_1\omega_1 \ldots \omega_i$, *a description of* ω, *and an additional description of at most* c *bits, we can reconstruct* n *by running* M *and observing only acceptance or rejection.*

(ii) *Given pushdown store contents after processing input* $\zeta_m \ldots \zeta_1\omega_1 \ldots \omega_n$ *and* M*'s state, we can reconstruct* w *from an additional description of at most* c *bits.*

(iii) $C(\omega_1 \ldots \omega_n) \geq 2 \log\log m$.

Then there is a constant c' *depending only on* L *and* c *such that* $C(w) \leq c'$.

Proof. Let L be accepted by M with input head h_r. Assume that m, n, w satisfy the conditions in the statement of the lemma. For convenience we write

$$u = \zeta_m \ldots \zeta_1, \qquad v = \omega_1 \ldots \omega_n.$$

For each input $z \in \Sigma^*$, we denote with $c(z)$ the pushdown store contents at the time h_r has read all of z, and moves to the right adjacent input symbol. Consider the computation of M on input uv from the time when h_r reaches the end of u. There are two cases, which follow.

Case 1. There is a constant c_1 such that for infinitely many pairs m, n that satisfy the statement of the lemma, if h_r continues and reaches the end of v, then all of the original $c(u)$ has been popped except at most the bottom c_1 bits.

That is, machine M decreases its pushdown store from size $l(c(u))$ to size c_1 during the processing of v. The first time this occurs, let v' be the processed initial segment of v, and v'' the unprocessed suffix (so that $v = v'v''$) and let M be in state q. We can describe w by the following items:[2]

- A self-delimiting description of M (including Σ) and this discussion in $O(1)$ bits.
- A self-delimiting description of ω in $(1+\epsilon)c$ bits.
- A description of $c(uv')$ and q in $c_1 \log|\Sigma| + O(1)$ bits.
- The "additional description" mentioned in item (i) of the statement of the lemma in self-delimiting format, using at most $(1+\epsilon)c$ bits. Denote it by p.
- The "additional" description mentioned in item (ii) of the statement of the lemma in self-delimiting format, using at most $(1+\epsilon)c$ bits. Denote it by r.

By item (i) in the statement of the lemma we can reconstruct v'' from M in state q and with pushdown store contents $c(uv')$, and ω, using description p. Subsequently, by starting M in state q with pushdown store contents $c(uv')$, we process v''. At the end of the computation we have obtained M's state and pushdown store contents after processing uv. According to item (ii) in the statement of the lemma, together with description r, we can now reconstruct w. Since $C(w)$ is at most the length of this description,

$$C(w) \le 4c + c_1 \log|\Sigma| + O(1).$$

Setting $c' := 4c + c_1 \log|\Sigma| + O(1)$ satisfies the lemma.

Case 2. By way of contradiction, assume that Case 1 does not hold. That is, for each constant c_1 all but finitely many pairs m, n that satisfy the conditions in the lemma cause M not to decrease its stack height below c_1 during the processing of the v part of input uv.

Fix some constant c_1. Set m, n so that they satisfy the statement of the lemma, and to be as long as required to validate the argument below. Choose u' as a suffix of $yy \dots yx$ with $l(u') > 2^m$ and

$$C(l(u')) < \log\log m. \tag{2}$$

That is, $l(u')$ is much larger than $l(u)$ $(= m)$ and much more regular. A moment's reflection learns that we can always choose such a u'.

CLAIM 4.3. For large enough m there exists a u' as above, such that M starts in the same state and accesses the same top $l(c(u)) - c_1$ elements of its stack during the processing of the v parts of both inputs uv and $u'v$.

Proof. By assumption, M does not read below the bottom c_1 symbols of $c(u)$ while processing the v part of input uv.

We argue that one can choose u' such that the top segment of $c(u')$ is precisely the same as the top segment of $c(u)$ above the bottom c_1 symbol for large enough $l(u)$, $l(u')$.

To see this we examine the initial computation of M on u. Since M is deterministic, it must either cycle through a sequence of pushdown store contents, or increase its pushdown store with repetitions on long enough u (and u'). Namely, let a triple (q, i, s) mean that M is in state q, has top pushdown store symbol s, and h_r is at ith bit of some y. Consider only the triples (q, i, s) at the steps where M will never go below the current top pushdown store level

[2] Since we need to glue together different binary items in the encoding, and in a way so that we can effectively separate them again, like $\langle x, y\rangle = x'y$, we count $C(x) + 2\log C(x) + 1$ bits for a self-delimited encoding $x' = 1^{l(l(x))}0l(x)x$ of x . We only need to give self-delimiting forms for all but one constituent description item.

again while reading u. (That is, s will not be popped before going into v.) There are precisely $l(c(u))$ such triples. Because the input is repetitious and M is deterministic, some triple must start to repeat within a constant number of steps and with a constant interval (in height of M's pushdown store) after M starts reading y's. It is easy to show that within a repeating interval only a constant number of y's are read.

The pushdown store does not cycle through an a priori bounded set of pushdown store contents, since this would mean that there is a constant c_1 such that the processing by M of any suffix of $yy\dots yx$ does not increase the stack height above c_1. This situation reduces to Case 1 with $v=\epsilon$.

Therefore, the pushdown store contents grow repetitiously and unboundedly. Since the repeating cycle starts in the pushdown store after a constant number of symbols, and its size is constant in number of y's, we can adjust u' so that M starts in the same state and reads the same top segments of $c(u)$ and $c(u')$ in the v parts of its computations on uv and $u'v$. This proves the claim. □

The following items form a description from which we can reconstruct v.

- This discussion and a description of M in $O(1)$ bits.
- A self-delimiting description of the recursive sequence ω of which v is an initial segment in $(1+\epsilon)c$ bits.
- A self-delimiting description of the pair $\langle x, y\rangle$ in $(1+\epsilon)c$ bits.
- A self-delimiting description of $l(u')$ in $(1+\epsilon)C(l(u'))$ bits.
- A program p to reconstruct v given ω and M's state and pushdown store contents after processing u. By item (i) of the statement of the lemma, $l(p)\le c$. Therefore, a self-delimiting description of p takes at most $(1+\epsilon)c$ bits.

The following procedure reconstructs v from this information. By using the description of M and u' we construct the state $q_{u'}$ and pushdown store contents $c(u')$ of M after processing u'. By Claim 4.3, the state q_u of M after processing u satisfies $q_u=q_{u'}$ and the top $l(c(u))-c_1$ elements of $c(u)$ and $c(u')$ are the same. Run M on input ω starting in state $q_{u'}$ and with stack contents $c(u')$. By assumption, no more than $l(c(u))-c_1$ elements of $c(u')$ get popped before we have processed $\omega_1\dots\omega_n$. By just looking at the consecutive states of M in this computation, and using program p, we can find n according to item (i) in the statement of the lemma. To reconstruct v requires by definition at least $C(v)$ bits. Therefore,

$$\begin{aligned} C(v) &\le (1+\epsilon)C(l(u'))+4c+O(1)\\ &\le (1+\epsilon)\log\log m+4c+O(1), \end{aligned}$$

where the last inequality follows by equation (2). But this contradicts item (iii) in the statement of the lemma for large enough m. □

Items (i)–(iii) in the KC-DCFL lemma can be considerably weakened, but the presented version gives the essential idea and power: it suffices for many examples. A more restricted, but easier, version is the following.

COROLLARY 4.4. *Let $L\subseteq\Sigma^*$ be a dcfl and let c be a constant. Let x and y be fixed finite words over Σ and let ω be a recursive sequence over Σ. Let u be a suffix of $yy\dots yx$, let v be a prefix of ω, and let $w\in\Sigma^*$ such that*

(i) *v can be described in c bits given L_u in lexicographical order*;

(ii) *w can be described in c bits given L_{uv} in lexicographical order*; *and*

(iii) $C(v)\ge 2\log\log l(u)$.

Then there is a constant c' depending only on L, c, x, y, ω such that $C(w)\le c'$.

All the following context-free languages were proved to be not dcfl only with great effort before [6], [4], [24]. Our new proofs are more direct and intuitive. Basically, if v is the first word in L_u, then processing the v part of input uv must have already used up the information of u. But if there is not much information left on the pushdown store, then the first word w in L_{uv} cannot have high Kolmogorov complexity.

EXAMPLE 9 (Exercise 10.5 (a)** in [6]). Prove $L = \{x : x = x^R, x \in \{0, 1\}^*\}$ is not dcfl. Suppose the contrary. Set $u = 0^n 1$ and $v = 0^n$, $C(n) \geq \log n$, which satisfies item (iii) of the lemma. Since v is lexicographically the first word in L_u, item (i) of the lemma is satisfied. The lexicographically first nonempty word in L_{uv} is 10^n, and so we can set $w = 10^n$ which satisfies item (ii) of the lemma. But now we have $C(w) = \Omega(\log n)$, which contradicts the KC-DCFL lemma and its corollary.

Approximately the same proof shows that the context-free language $\{xx^R : x \in \Sigma^*\}$ and the context-sensitive language $\{xx : x \in \Sigma^*\}$ are not deterministic context-free languages.

EXAMPLE 10 (Exercise 10.5 (b)** in [6] and Example 1 in [24]). Prove $\{0^n 1^m : m = n, 2n\}$ is not dcfl. Suppose the contrary. Let $u = 0^n$ and $v = 1^n$, where $C(n) \geq \log n$. Then v is the lexicographically first word in L_u. The lexicographically first nonempty word in L_{uv} is 1^n. Set $w = 1^n$, and $C(w) = \Omega(\log n)$, contradicting the KC-DCFL lemma and its corollary.

EXAMPLE 11 (Example 2 in [24]). Prove $L = \{xy : l(x) = l(y), y$ contains a "1," $x, y \in \{0, 1\}^*\}$ is not dcfl. Suppose the contrary. Set $u = 0^n 1$ where $l(u)$ is even. Then $v = 0^{n+1}$ is lexicographically the first even length word not in L_u. With $C(n) \geq \log n$, this satisfies items (i) and (iii) of the lemma. Choosing $w = 10^{2n+3}$, the lexicographically first even length word not in L_{uv} starting with a "1", satisfies item (ii). But $C(w) = \Omega(\log n)$, which contradicts the KC-DCFL lemma and its corollary.

EXAMPLE 12. Prove $L = \{0^i 1^j 2^k : i, j, k \geq 0, i = j \text{ or } j = k\}$ is not dcfl. Suppose the contrary. Let $u = 0^n$ and $v = 1^n$, where $C(n) \geq \log n$, satisfying item (iii) of the lemma. Then, v is lexicographically the first word in L_u, satisfying item (i). The lexicographic first word in $L_{uv} \cap \{1\}\{2\}^*$ is 12^{n+1}. Therefore, we can set $w = 12^{n+1}$ and satisfy item (ii). Then $C(w) = \Omega(\log n)$, contradicting the KC-DCFL lemma and its corollary.

EXAMPLE 13 (pattern-matching). The KC-DCFL lemma and its corollary can be used in a tricky manner. We prove $\{x\#yx^R z : x, y, z \in \{0, 1\}^*\}$ is not dcfl. Suppose the contrary. Let $u = 1^n\#$ and $v = 1^{n-1}0$, where $C(n) \geq \log n$, which satisfies item (iii) of the lemma. Since $v' = 1^n$ is the lexicographically first word in L_u, the choice of v satisfies item (i) of the lemma. (We can reconstruct v from v' by flipping the last bit of v' from 1 to 0.) Then $w = 1^n$ is lexicographically the first word in L_{uv}, to satisfy item (ii). Since $C(w) = \Omega(\log n)$, this contradicts the KC-DCFL lemma and its corollary.

5. Recursive, recursively enumerable, and beyond. It is immediately obvious how to characterize recursive languages in terms of Kolmogorov complexity. If $L \subseteq \Sigma^*$, and $\Sigma^* = \{v_1, v_2, \ldots\}$ is effectively ordered, then we define the characteristic sequence $\lambda = \lambda_1, \lambda_2, \ldots$ of L by $\lambda_i = 1$ if $v_i \in L$ and $\lambda_i = 0$ otherwise. In terms of the earlier developed terminology, if A is the automaton accepting L, then λ is the characteristic sequence associated with the equivalence class $[\epsilon]$. Recall Definition 4.1 of a recursive sequence. A set $L \in \Sigma^*$ is recursive iff its characteristic sequence λ is a recursive sequence. The next theorem then follows from the definitions and the first paragraph of the Appendix.

THEOREM 5.1 (recursive KC characterization). *A set $L \in \Sigma^*$ is recursive iff there exists a constant c_L (depending only on L) such that, for all n, $C(\lambda_{1:n}|n) < c_L$.*

L is r.e. (recursively enumerable) if the set $\{n : \lambda_n = 1\}$ is r.e. In terms of Kolmogorov complexity, the following theorem gives not only a qualitative but even a quantitative difference between recursive and r.e. languages. The following theorem is due to Barzdin' [1], [13].

THEOREM 5.2 (KC r.e.). (i) *If L is r.e., then there is a constant c_L (depending only on L), such that for all n, $C(\lambda_{1:n}|n) \leq \log n + c_L$.*

(ii) *There exists an r.e. set L such that $C(\lambda_{1:n}) \geq \log n$, for all n.*

Note that, with L as in item (ii), the set $\Sigma^* - L$ (which is possibly non-r.e.) also satisfies item (i). Therefore, item (i) is not a Kolmogorov complexity characterization of the r.e. sets.

EXAMPLE 14. Consider the standard enumeration of Turing machines. Define $k = k_1 k_2 \ldots$ by $k_i = 1$ if the ith Turing machine started on its ith program halts ($\phi_i(i) < \infty$), and $k_i = 0$ otherwise. Let A be the language such that k is its characteristic sequence. Clearly, A is an r.e. set. In [1] it is shown that $C(k_{1:n}) \geq \log n$ for all n.

EXAMPLE 15. Let k be as in the previous example. Define a one-way infinite binary sequence h by

$$h = k_1 0^2 k_2 0^{2^2} \ldots k_i 0^{2^i} k_{i+1} \ldots .$$

Then, $C(h_{1:n}) = O(C(n)) + \Theta(\log \log n)$. Therefore, if h is the characteristic sequence of a set B, then B is not recursive, but more "sparsely" nonrecursive than A is.

EXAMPLE 16. The probability that the optimal universal Turing machine U halts on self-delimiting binary input p, randomly supplied by tosses of a fair coin, is $\Omega, 0 < \Omega < 1$. Let the binary representation of Ω be $0.\Omega_1 \Omega_2 \ldots$ Let Σ be a finite nonempty alphabet, and $v_1, v_2, \ldots$ an effective enumeration without repetitions of Σ^*. Define $L \subseteq \Sigma^*$ such that $v_i \in L$ iff $\Omega_i = 1$. It can be shown (see, for example, [12]) that the sequence $\Omega_1, \Omega_2, \ldots$ satisfies

$$C(\Omega_{1:n}|n) \geq n - \log n - 2 \log \log n - O(1),$$

for all but finitely many n.

Hence neither L nor $\Sigma^* - L$ is r.e. It is not difficult to see that $L \in \Delta_2 - (\Sigma_1 \cup \Pi_1)$, in the arithmetic hierarchy (that is, L is not recursively enumerable) [22], [23].

6. Questions for future research. (a) It is not difficult to give a direct KC analogue of the $uvwxy$ pumping lemma (as Tao Jiang pointed out to us). Just like the pumping lemma, this will show that $\{a^n b^n c^n : n \geq 1\}$, $\{xx : x \in \Sigma^*\}$, $\{a^p : p \text{ is prime}\}$, and so on, are not cfl. Clearly, this hasn't yet captured the Kolmogorov complexity heart of cfl. In general, can we find a CFL-KC characterization?

(b) What about ambiguous context-free languages?

(c) What about context-sensitive languages and deterministic context-sensitive languages?

Appendix: Proof of Claim 3.8. A *recursive real* is a real number whose binary expansion is recursive in the sense of Definition 4.1. The following result is demonstrated in [14] and attributed to A.R. Meyer. For each constant c there are only finitely many $\omega \in \{0, 1\}^\infty$ with $C(\omega_{1:n}|n) \leq c$ for all n. Moreover, each such ω is a recursive real.

In [2] this is strengthened to a version with $C(\omega_{1:n}) \leq C(n) + c$, and strengthened again to a version with $C(\omega_{1:n}) \leq \log n + c$. Claim 3.8 is weaker than the latter version by not requiring the ω's to be recursive reals. For completeness sake, we present a new direct proof of Claim 3.8 avoiding the notion of recursive reals.

Recall our convention of identifying integer x with the xth binary sequence in lexicographical order of $\{0, 1\}^*$ as in (1).

Proof of Claim 3.8. Let c be a positive constant, and let

$$\begin{aligned} A_n &= \{x \in \{0, 1\}^n : C(x) \leq \log n + c\}, \\ A &= \{\omega \in \{0, 1\}^\infty : \forall_{n \in \mathcal{N}}[C(\omega_{1:n}) \leq \log n + c]\} . \end{aligned} \tag{3}$$

If the cardinality $d(A_n)$ of A_n dips below a fixed constant c', for infinitely many n, then c' is an upper bound on $d(A)$. This is because it is an upper bound on the cardinality of the set of prefixes of length n of the elements in A, for *all* n.

Fix any $l \in \mathcal{N}$. Choose a binary string y of length $2l + c + 1$ that satisfies

$$C(y) \geq 2l + c + 1. \tag{4}$$

Choose i maximum such that for division of y in $y = mn$ with $l(m) = i$ we have

$$m \leq d(A_n). \tag{5}$$

(This holds at least for $i = 0 = m$.) Define similarly a division $y = sr$ with $l(s) = i + 1$. By maximality of i, we have $s > d(A_r)$. From the easily proven $s \leq 2m + 1$, it then follows that

$$d(A_r) \leq 2m. \tag{6}$$

We prove $l(r) \geq l$. Since by (5) and (3) we have

$$m \leq d(A_n) \leq 2^c n,$$

it follows that $l(m) \leq l(n) + c$. Therefore,

$$2l + c + 1 = l(y) = l(n) + l(m) \leq 2l(n) + c,$$

which implies that $l(n) > l$. Consequently, $l(r) = l(n) - 1 \geq l$.

We prove $d(A_r) = O(1)$. By dovetailing the computations of the reference universal Turing machine U for all programs p with $l(p) \leq \log n + c$, we can enumerate all elements of A_n. We can reconstruct y from the mth element, say y_0, of this enumeration. Namely, from y_0 we reconstruct n since $l(y_0) = n$, and we obtain m by enumerating A_n until y_0 is generated. By concatenation we obtain $y = mn$. Therefore,

$$C(y) \leq C(y_0) + O(1) \leq \log n + c + O(1). \tag{7}$$

From (4), we have

$$C(y) \geq \log n + \log m. \tag{8}$$

Combining (7) and (8), it follows that $\log m \leq c + O(1)$. Therefore, by (6),

$$d(A_r) \leq 2^{c+O(1)}.$$

Here, c is a fixed constant independent of n and m. Since $l(r) \geq l$ and we choose l arbitrarily, $d(A_r) \leq c_0$ for a fixed constant c_0 and infinitely many r, which implies $d(A) \leq c_0$, and hence the claim. □

We avoided establishing, as in the cited references, that the elements of A defined in (3) are recursive reals. The resulting proof is simpler, and sufficient for our purpose, since we only need to establish the finiteness of A.

REMARK 3. The difficult part of the Regular KC Characterization theorem above consists in proving that the KC Regularity lemma is exhaustive, i.e., can be used to prove the nonregularity of all nonregular languages. Let us look a little more closely at the set of sequences defined in item (iii) of the KC Characterization theorem. The set of sequences A of (3) is a superset of the set of characteristic sequences associated with L. According to the proof in the cited references, this set A contains finitely many *recursive* sequences (computable by Turing machines). The subset of A consisting of the characteristic sequences associated with L, satisfies much more stringent computational requirements, since it can be computed using only the finite automaton recognizing L. If we replace the plain Kolmogorov complexity in the statement of the theorem by the so-called "prefix complexity" variant K, then the equivalent set of A in (3) is

$$\{\omega \in \{0, 1\}^\infty : \forall_{n \in \mathcal{N}}[K(\omega_{1:n}) \leq K(n) + c]\},$$

which is finite [12, Exercise 3.24] and contains nonrecursive sequences by a result of Solovay [20].

Acknowledgments. We thank Peter van Emde Boas, Theo Jansen, Tao Jiang for reading the manuscript and commenting on it, and the anonymous referees for extensive comments and suggestions for improvements. John Tromp improved the proof of Claim 3.8.

REFERENCES

[1] Y.M. BARZDIN', *Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set*, Soviet Math. Dokl., 9 (1968), pp. 1251–1254.

[2] G.J. CHAITIN, *Information-theoretic characterizations of recursive infinite strings*, Theoret. Comput. Sci., 2 (1976), pp. 45–48.

[3] A. EHRENFEUCHT, R. PARIKH, AND G. ROZENBERG, *Pumping lemmas for regular sets*, SIAM J. Comput., 10 (1981), pp. 536–541.

[4] M.A. HARRISON, *Introduction to Formal Language Theory*, Addison-Wesley, Reading, MA, 1978.

[5] D.R. HEATH-BROWN AND H. IWANIEC, *The difference between consecutive primes*, Invent. Math., 55 (1979), pp. 49–69.

[6] J.E. HOPCROFT AND J.D. ULLMAN, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.

[7] J. JAFFE, *A necessary and sufficient pumping lemma for regular languages*, SIGACT News, 10 (1978), pp. 48–49.

[8] T. JIANG AND M. LI, *k one-way heads cannot do string matching*, Proc. 25th ACM Symp. Theory of Computing, ACM Press, 1993, pp. 62–70.

[9] T. JIANG, J.I. SEIFERAS, AND P. VITÁNYI, *Two heads are better than two tapes*, Proc. 26th ACM Symp. Theory of Computing, ACM Press, 1994, pp. 668–675.

[10] A.N. KOLMOGOROV, *Three approaches to the quantitative definition of information*, Problems Inform. Transmission, 1 (1965), pp. 1–7.

[11] M. LI AND P.M.B. VITÁNYI, *Tape versus queue and stacks: The lower bounds*, Inform. and Comput., 78 (1988), pp. 56–85.

[12] ———, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York, 1993.

[13] D.W. LOVELAND, *On minimal-program complexity measures*, in Proc. (1st) ACM Symp. Theory of Computing, ACM Press, pages 61–65, 1969.

[14] ———, *A variant of the Kolmogorov concept of complexity*, Inform. and Control, 15 (1969), pp. 510–526.

[15] W. MAASS, *Combinatorial lower bound arguments for deterministic and nondeterministic Turing machines*, Trans. Amer. Math. Soc., 292 (1985), pp. 675–693.

[16] P. MARTIN-LÖF, *The definition of random sequences*, Inform. and Control, 9 (1966), pp. 602–619.

[17] ———, *Complexity oscillations in infinite binary sequences*, Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete, 19 (1971), pp. 225–230.

[18] W. J. PAUL, *Kolmogorov's complexity and lower bounds*, in L. Budach, ed., Proc. 2nd International Conference on Fundamentals of Computation Theory, pp. 325–334, Akademie Verlag, Berlin, 1979.

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

[20] R. SOLOVAY, lecture notes, unpublished, University of California at Los Angeles, 1975.

[21] D. STANATO AND S. WEISS, *A pumping theorem for regular languages*, SIGACT News, 14 (1982), pp. 36–37.

[22] M. VAN LAMBALGEN, *Random Sequences*, Ph.D. thesis, Universiteit van Amsterdam, Amsterdam, 1987.

[23] ———, *Algorithmic information theory*, J. Symbolic Logic, 54 (1989), pp. 1389–1400.

[24] S. YU, *A pumping lemma for deterministic context-free languages*, Inform. Process. Lett., 31 (1989), pp. 47–51.