A New Approach to Formal Language Theory by Kolmogorov Complexity
(Preliminary Version)

Ming Li

York University, Department of Computer Science
North York, Ontario M3J 1P3, Canada

Paul M.B. Vitanyi

Centrum voor Wiskunde en Informatica
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
and
Universiteit van Amsterdam, Faculteit Wiskunde en Informatica

ABSTRACT

We introduce Kolmogorov complexity as a new technique in Formal Language Theory. We give an alternative for pumping lemma(s) and a new characterization for regular languages. For the separation of deterministic contextfree languages and contextfree languages no pumping lemmas or any other general method was known. We give a first general technique to separate these classes, and illustrate its use on four examples previously requiring labourous ad hoc methods. The approach is also successful at the high end of the Chomsky hierarchy since one can quantify nonrecursiveness in terms of Kolmogorov complexity. We also give a new proof, using Kolmogorov complexity, of Yao and Rivest's result that $k + 1$ heads are better than k heads.

1. Introduction

It is feasible to reconstruct parts of Formal Language Theory using algorithmic information theory (Kolmogorov complexity). We prove theorems on how to use Kolmogorov complexity as a concrete, powerful, tool. We do not just want to introduce fancy mathematics; our goal is to help our readers do proofs in the most essential, usually easiest, 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 of the type that are used as a tool. Once derived, our theorems are very easy to apply.

1.1. Prelude

The first application of Kolmogorov complexity in the theory of computation was in [16, 17]. By re-doing 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 [10, 14]. See also a survey of two decades of applied Kolmogorov complexity [11].

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'. A Kolmogorov random string cannot be (effectively) compressed. Incompressibility is a noneffective property: it can be shown that no particular string, except finitely many, can be proved to be random. 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 a particular "typical" input is *hard* (sometimes impossible) to find effectively, the proof has to involve all the inputs. Now we understand that a "typical input" can be constructed via a Kolmogorov random string. But we cannot exhibit such a typical input since we cannot prove a string to be random. No wonder the old counting arguments had to involve all inputs. In a proof using Kolmogorov complexity, we first choose a random string that is known to *exists* (although we cannot construct it). Then we show that if the assumed lower time bound would not hold, then this string could be compressed, and hence it would not be random.

1.2. Outline of This Paper

It turns out that the same approach works in a new area for application of Kolmogorov complexity: Formal Languages and Automata Theory proper. We show a powerful alternative to the pumping lemma 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, see for instance [6]. Comparatively recently, a pumping lemma (to end all pumping lemmas) was exhibited: namely a pumping lemma that characterizes the regular languages [4]. This ultimate pumping lemma is complicated and uses Ramsey theory. In contrast, using Kolmogorov complexity we give a new characterization of the regular languages that simply makes our intuition of "finite state"ness of these languages rigorous and is easy to apply. Being a characterization it works for all non-regular languages. We give several examples of its application, some of which were quite difficult using pumping lemmas.

While there is a pumping lemma to show that certain languages are not context-free (CFL), there is *no* pumping lemma or any other general technique to separate the deterministic contextfree languages (DCFL) from the CFL languages. All known examples required very laborous ad hoc proofs, cf. [6]. We give necessary (Kolmogorov complexity) conditions for DCFL, that easily separates CFL from DCFL on all witness languages we have tried. We test the new method on four examples, which were very hard to handle before. For completeness we present a known characterization of recursive languages, and a necessary condition for recursively enumerable languages.

As further examples of the Kolmogorov complexity approach we test the approach on some known results: deterministic machines equivalent to nondeterministic machines may require exponentially more states, and a new proof of the Yao-Rivest result that $k+1$ heads are better than k heads for multihead finite state automata. We include some exercises for the reader.

2. Kolmogorov Complexity

Any of the usual definitions of Kolmogorov complexity [2, 8, 11, 17] will do for the sequel. To fix thoughts, consider the problem of describing a string x over 0's and 1's. Any partial recursive function f from strings over 0's and 1's to such strings, together with a string p, the *program* for f to compute x, such that $f(p) = x$, is such a description. It is useful to generalize this idea to the conditional version: $f(p, x) = y$ such that p is a program for f to compute x given y. Then the *descriptional* complexity K_f of x, *relative* to f and y, is defined by

$$K_f(x \mid y) = \min\{ |p| : p \in \{0,1\}^*,\ f(p,y) = x \},$$

or ∞ if no such p exists, where $|x|$ is the *length* (number of bits) of string x.

For a *universal* partial recursive function f_0 we know that, for all f, there is a constant c_f such that for all strings x, y, $K_{f_0}(x \mid y) \leqslant K_f(x \mid y) + c_f$. So the *canonical relative descriptional* complexity $K(x \mid y)$ can be set equal to $K_{f_0}(x \mid y)$. Hence we fix a reference universal function f_0 and dispense with the subscript: the *conditional Kolmogorov complexity* of x given y is defined as $K(x \mid y) = K_{f_0}(x \mid y)$, and the *unconditional Kolmogorov complexity* of x is $K(x) = K(x \mid \epsilon)$, where ϵ denotes the empty string ($|\epsilon| = 0$).

Since there are 2^n binary strings of length n, but only $2^n - 1$ possible shorter descriptions d, it follows that $K(x) \geqslant |x|$ for some binary string x of each length. We call such strings *incompressible* or *random*. It also follows that, for any length n and any binary string y, there is a binary string x of length n such that $K(x \mid y) \geqslant |x|$.

Example. (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 $K(v)$, a description of $|u|$, 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, as explained in the next section. Together this takes $K(v) + |uw| + O(\log |x|)$ bits. Hence,

$$K(x) \leqslant K(v)+O(\log |x|)+|uw| \quad ,$$

Thus, if we choose x incompressible, $K(x) \geqslant |x|$, then we obtain

$$K(v) \geqslant |v| - O(\log |x|) \quad .$$

It can be shown that this is optimal - a substring of an incompressible string can be compressible. This conforms to a fact we know from probability theory: every sufficiently long random string must contain long runs of zeros, so it must contain some substring which is compressible.

Example. (State Blow-Up in Converting NFA to DFA) The usual construction to convert a nondeterministic finite automaton (NFA) with n states into a deterministic finite automaton (DFA) with $f(n)$ states that recognizes the same language uses $f(n) = O(2^n)$ [9]. No general construction exists that uses significantly less states. We provide a new proof of this.

To convert an n-state NFA to a DFA, the DFA requires sometimes $2^{\Omega(n)}$ *states.*

Proof. Consider the language $L_k = \{x \mid$ the kth bit from right is $1\}$. L_k can be accepted by a *NFA* with $k+1$ states. Suppose a DFA A with only $2^{o(k)}$ states accepts L_k. We will also use A to denote A's description. Fix a string x of length k such that $K(x \mid A,k) \geqslant |x|$. Give $x0^*$ to A as input. Stop A when it reads the last bit of x. Record the current state of A. Reconstruct x by running A starting from the recorded current state; Feed A with input 0's; At the ith 0, if A accepts, then the ith bit of x is 1, otherwise it is 0. This description of x needs only $o(k)$ bits since A has only $2^{o(k)}$ states. So $K(x \mid A,k) < |x|$, contradiction. ●

Corollary. (Language reversal) The same argument shows that if L is accepted by an n-state *DFA*, then any *DFA* accepting L^R (L reversed) may require $2^{\Omega(n)}$ states.

Corollary. (One-way input versus two-way input) The same argument shows that if L is accepted by an n-state 2-way *DFA*, then any one-way *DFA* accepting L may require $2^{\Omega(n)}$ states.

Example. (Descriptions and Self-Delimiting Strings) Let x be a binary string of length n. The shortest program (and its length $K(x)$) of x is generally uncomputable. Let us consider 'good' computable approximations to it. Now a *description* of x can be given as follows.

(1) A piece of text containing several formal parameters $p_1, \ldots, p_m$. Think of this piece of text as a formal parametrized procedure in an algorithmic language like PASCAL. It is followed by

(2) an ordered list of the actual values of the parameters.

The piece of text of (1) can be thought of as being encoded over a given finite alphabet, each symbol of which is coded in bits. Therefore, the encoding of (1) as prefix of the binary description of x requires $O(1)$ bits. This prefix is followed by the ordered list (2) of the actual values of $p_1, \ldots, p_m$ in binary. To distinguish one from the other, we encode (1) and the different items in (2) as self-delimiting strings, as follows.

For natural numbers x, in the sequel of this paper let x denote both the natural number and the xth binary string in the sequence 0, 1, 00, 01, 10, 11, 000,.... So the natural number 3 corresponds both to the natural number 3 and to the binary string 00. For each string x, the string $\bar{x}$ is obtained by inserting a zero after each letter in x except for the last letter where we insert a one. Let $x' = \overline{|x|}\, w$. The string x' is called the *self-delimiting* code of x. So '100101011' is the self-delimiting code of '01011'. The self-delimiting code of a positive integer x requires $\log x + 2\log\log x$ bits, which is equivalent to saying that the self-delimiting code of a binary string x requires $|x| + 2\log|x|$ bits. All logarithms are base 2 unless otherwise noted. For convenience, we denote the length $|x|$ of a natural number n by "$\log x$".

3. Regular Sets and Finite Automata

It is useful to first develop formal language theory in a way that is not exactly new, but for some reason has fallen into disregard. In connection with the Kolmogorov approach, we believe that a simple and transparent theory results.

An automaton is a 'black box' function $B: V \times M \rightarrow M$, with V the nonempty finite input alphabet, M a set of states of memory. We extend B to B' on V^* by $B'(\epsilon, m) = m$ and

$$B'(a(1)...a(n), m) = B(a(n), B'(a(1)...a(n-1), m)) .$$

Clearly, if B' is not one-one, then the automaton 'forgets' because some x and y from V^* drive B into the same memory state. Assuming an initial memory state m_0, we denote 'indistinguishability' of a pair of histories $x,y \in V^*$ by $x \sim y$, defined as $B'(x,m_0) = B'(y,m_0)$. 'Indistinguishability' of objects is intuitively reflexive, symmetric, transitive, and right-invariant, i.e., $\sim$ is a right-invariant equivalence relation on V^*. It is a simple matter to ascertain this formally. We add output by a function $o: M \rightarrow \{0,1\}$, and say B accepts a language L defined as $\{x \in V^* : B'(x,m_0) = m,\ o(m) = 1\}$. If the set of classes induced by $\sim$ is finite, then B is a *finite automaton*. This way, it is a straight-forward exercise to verify from the definitions:

Theorem (Myhill, Nerode). *The following statements about $L \subseteq V^*$ are equivalent.*
(i) $L \subseteq V^*$ is accepted by some finite automaton.
(ii) L consists of a union of right-invariant $\sim$-equivalence classes of V^*, where $|V^*/\sim| < \infty$.
(iii) For all $x,y \in V^*$ define $x \sim y$ by: for all $z \in V^*$ we have $xz \in L$ iff $yz \in L$. Then $|V^*/\sim| < \infty$.

Subsequently, closure under complement, union and intersection follow by simple construction of the appropriate black box functions from given ones. 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, e.g., the *uvw*-lemma. However, these lemmas are quite difficult to state and cumbersome to prove or use. In contrast, below we show how to replace such arguments by simple, intuitive and yet rigorous, Kolmogorov complexity arguments. Below, languages are infinite sets of strings over a finite alphabet.

Regular languages coincide with the languages accepted by finite automata (FA). 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 the accepting FA after processing 0^k is, up to a constant, a description of k. Namely, by running the FA, starting from state q, on a string consisting of 1's, it reaches its first accepting state precisely after k 1's. Hence, since the FA 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. (In the lexicographic order short strings precede long strings.)

Lemma (KC-Regularity). *Let L be regular, and c a constant depending only on L. For each x, if xy is the nth string in the lexicographical order in (or in the complement of) $L_x = \{xy : xy \in L\}$ then $K(y) \leq K(n) + c$.*

Proof. Let L be a regular language. A 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,
- 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 another application of the KC-Regularity Lemma we prove that $\{1^p : p$ is prime $\}$ is not regular. Consider the string xy consisting of p 1's, with p is the $(k+1)$th prime. Set in the lemma x equal to $1^{p'}$ with p' the kth prime, so $y = 1^{p-p'}$, and $n = 1$. It follows that $K(p - p') = O(1)$. Since the differences between the consecutive primes rise unbounded, this implies that there is an unbounded number of integers of Kolmogorov complexity $O(1)$. Since there are only $O(1)$ descriptions of length $O(1)$, we have a contradiction. (To prove that $p - p'$ rises unbounded: If P is the product of the first j primes, then no $P + i$ $(1 \leq i \leq j)$ is prime.) We give two more examples from the well-known textbook of Hopcroft and Ullman that are marked * as difficult there:

Example [Exercise 3.1(h)* in [6]]. Prove that $L=\{xx^Rw : x,w \in \{0,1\}^*\}$ is not regular. Fix x such that $K(x) \geqslant |x|$. Consider prefix $(01)^{3\log|x|}x$. The first string with this prefix in L is $(01)^{3\log|x|}xx^R(10)^{3\log|x|}0$. By the KC-regularity lemma, $K(x^R(10)^{3\log|x|}0) \leqslant K(1)+c$, a contradiction.

Example [Exercise 3.6* in [6]]. Prove that $L=\{0^i1^j : GCD(i,j)=1\}$ is not regular. Obviously L is regular iff $L'=\{0^i1^j : GCD(i,j) \neq 1\}$ is regular. Fix a prime p such that $K(p) \geqslant \log p - \log\log p$ (by density of primes). Consider prefix 0^p. By the KC-regularity lemma, $K(1^p) \leqslant K(2)+c$, a contradiction.

3.2. Kolmogorov Complexity Characterization of Regular Languages

We can show that the lemma is not only a device to show that some nonregular languages are nonregular, as the common pumping lemmas, but the condition is a characterization of the regular sets. (So it can be used to show nonregularity for all nonregular languages.) While the pumping lemma's are not precise enough (except for the difficult Ehrenfeucht-Parikh-Rozenberg construction) to characterize the regular languages, with Kolmogorov complexity this is easy. In fact, the lemma above is a direct corrollary of the characterization below. If V is a finite nonempty alphabet, then fix an effective order $v_1, v_2, \cdots$ of the elements of V^*. (This can be the lexicographic order.) For each $x \in V^*$, let $\chi = \chi_1\chi_2 \cdots$ be the *characteristic sequence* of x, such that the ith element $\chi_i = 1$ if $xv_i \in L$, and $\chi_i = 0$ otherwise. We denote $\chi_1 \cdots \chi_n$ by $\chi_{1:n}$.

Theorem (Regular KC-Characterization). *Let $L \subseteq V^*$. The following statements are equivalent.*

(i) L is regular.

(ii) There is a constant c_L depending only on L, such that for all $x \in V^*$, for all n, $K(\chi_{1:n} \mid n) < c_L$.

(iii) There is a constant c_L depending only on L, such that for all $x \in V^*$, for all n, $K(\chi_{1:n}) < K(n) + c_L$.

Proof (Outline). (i)→(ii): by similar proof as the KC-Regularity Lemma.

(ii)→(iii): obvious.

(iii)→(i): Define χ is *recursive*, if there is a recursive function $f : N \to \{0,1\}$ such that $\chi_n = f(n)$ for all n.

Claim. For each constant c there are only finitely many χ such that, for all n, $K(\chi_{1:n}) \leqslant K(n) + c$, and each of these χ is recursive.

Proof. Omitted. It follows by combining arguments due to D.W. Loveland, A.R. Meyer and G.J. Chaitin in [3, 12]. □

By (iii) and the claim, there are only finitely many distinct χ's associated with the x's in V^*, and all of them are recursive. Define the right-invariant equivalence relation $\sim$ by $x \sim x'$ if $\chi = \chi'$. This relation induces a partition of V^* 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. □

The difficult part of the Regular KC-Characterization consists in proving that the KC-Regularity Lemma is exhaustive, i.e., can be used to prove the nonregularity of all nonregular languages. This is non-trivial, since Item (iii) does not hold for the self-delimiting version of Kolmogorov complexity.

Exercises

1. Prove that $\{0^n 1^m \mid m > n\}$ is not regular.

2. Prove that $\{x\#xy \mid x, y \in \{0, 1\}^*\}$ is not regular.

3. Prove that $\{x\#y \mid x \textit{ appears (possibly nonconsecutively) in } y\}$ is not regular.

4. Prove that $\{x\#y \mid \textit{at least } \frac{1}{2} \textit{ of } x \textit{ is a substring in } y\}$ is not regular.

5. Prove that $\{x\#y\#z \mid x^*y = z\}$ is not regular.

6. Prove that $\{p \mid p \textit{ is a } \text{prime} \textit{ represented in binary starting with a } 1\}$ is not regular.

4. Context-free Languages

In this section we study CFL's and DCFL's (deterministic context-free languages) using Kolmogorov complexity. We provide a lemma to establish necessary properties in terms of Kolmogorov complexity for a language to be DCFL. Our lemma can be used to prove many CFL languages to be non-DCFL's. This is all the more interesting, since there does not appear to be a natural pumping lemma to separate DCFL from CFL; previously the only recourse was to *ad hoc* reasoning.

4.1. Necessary Conditions for Deterministic Context-free Languages

While there are pumping lemmas to show nonregularity, we hope to have convinced the reader that using Kolmogorov complexity is both easier and more natural. To prove that a language is in CFL - DCFL there is no pumping lemma at all; yet in this section we present a KC-DCFLness Lemma that is easily used to demonstrate witnesses to the nonemptiness of CFL - DCFL. Previously, this was done one at a time in an *ad hoc* fashion. The resulting proofs were usually quite complicated. (See for example the Solution for Exercise 10.5 (a) in [6].)

For a string $x = x_1 x_2 \ldots x_n$, we use notation $x_{i:j} = x_i x_{i+1} \ldots x_j$. x^R is the reverse of x. We say that a string x is *finitely generatable* if x is a prefix of the infinite string generated by some finite state deterministic machine (on empty input). Like in the case of the regular sets, we first state a simplified version of the theorem we aim at. We also use the definitions and notions in the previous section. If M is a dpda, then we use $|M|$ to denote the length of a self-delimiting description of it.

Lemma (KC-DCFL). *Let L be a DCFL accepted by a dpda M, and let FS_u and FS_v be finite state generators. For large enough u and v such that uv is the first word in (or in the complement of) L with prefix u, and moreover*

(a) *u^R and v are finitely generatable by finite state generators FS_u and FS_v respectively; and*

(b) *$K(u) > \log|u|$, $K(v) > \log|v|$ and* $\log\log|u| < K(v)/2$;

we have that if uvw is the first word in (or in the complement of) L with proper prefix uv, then $K(w) = O(1)$.

Corollary (KC-DCFL). Above lemma also holds if $uv'w$ is the first word with proper prefix uv', where v' is obtained from v with the last (few) bit(s) of v being changed.

Proof Sketch. Let L be accepted by M with input head h_r and pushdown store head h_p. Assume uv, $uvw \in L$ and they satisfy the above conditions. (The case uv or uvw is an element of the complement of L is handled similarly, since the dpda recognizes both L and its complement.) For each x, we denote with $c(x)$ the pushdown store contents at the time h_r has read all of x, and moves to the right adjacent input. Consider time t when h_r reaches the end of u. There are two cases:

Case 1. Suppose that when h_r continues and reaches the end of v, all of the original $c(u)$ has been popped except the bottom C bits, where C is a constant not depending on u, v. If at the time the pushdown store first decreased from $|c(u)|$ to size C the input head h_r was at position p in v, then we must have $K(v_{p:|v|}) \leqslant C + |M| + O(1)$ $(= O(1))$. Namely, no word (lexicographically before) uv is in L, while uv is in L, and therefore $v_{p:|v|}$ can be reconstructed from the pushdown contents and a description of M. This implies that $K(c(uv)) = O(1)$. But, since uvw is the first string in L with proper prefix uv, we must have $K(w) = O(1)$, by the standard argument (since we can reconstruct w easily).

Case 2. Suppose $c(uv)$ still contains the bottom $f(u, v)$ bits of the original $c(u)$, where $f(u, v)$ is unbounded. We show that this contradicts assumption (b).

First generate a long "easy" u' with suffix u, using the same generating process FS_u that finitely generates u^R, such that $K(|u'|) < \log\log|u|$, but $|u'| >> |u|$. Then, u is a suffix of u' and $K(u') < \log\log|u| + O(1)$.

Claim. There is such a u' such that M accepts $u'v$, and M does not accept any prefix $u'v'$ of $u'v$.

Proof. Since u is a proper and very short suffix of u', we can choose u' such that the top segment of $c(u')$ to be read by M is precisely the same as the top segment of $c(u)$ to be read by M in the v-parts of its computations on inputs uv and $u'v$, for large enough $|u|$, $|u'|$. This follows from well-known arguments, related to the determinacy of both FA_u and M. To see it, notice that both u and u' are generated by the same finite state machine. Such a machine must generate the string of form $a^R(b^R)^\infty$ for constant size strings a,b. So $u = cb^ka$, where c is a suffix of b. Clearly, we can choose $u' = cb^{k'}a$ with $k' >> k$ and still $K(u') < \log\log|u| + O(1)$. Since M is deterministic, it must either cycle through a sequence of stack contents, or increase its stack with repetititions on long enough u (and u'). Namely, let a triple (q,i,s) mean that M is in state q, has top stack symbol s, and h_r is at ith bit of some b. Consider only the triples (q,i,s) at the steps where M will never go below the current top stack level again while reading u. (*I.e.* s will not be popped before going into v.) There are precisely $h = |c(u)|$ such triples. Because the input is repetitious and M deterministic, some triple must start to repeat within a constant number of steps and with a constant interval (in height of M's stack) after M starts reading b's. It is easy to show that within a repeating interval only a constant number of b's are read. The stack does not cycle through a finite set of stack contents, since $c(u) + |M| \geqslant K(v) \geqslant \log |v|$ (because we can reconstruct v from $c(u)$ and M). So the stack contents grows repetitious and unbounded. Since the repeating cycle starts in the stack after a constant number of symbols, and its size is constant in number of b's, it is easy to 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 part of its computation on uv and $u'v$. This proves the Claim. □

By the Claim, we can use u' and FS_v to find v. But this implies $K(v) \leqslant K(u') + O(1)$, and since $K(u') < \log\log|u| + O(1) < K(v)/2 + O(1)$, we have a contradiction again. This proves the Lemma. The Corollary follows by about the same proof. □

We next state the lemma in a more general form, sacrificing clarity. Assume the conceptual apparatus developed at the outset of Section 4, but this time the 'black box' function B is a dpda. This means that the 'indistinguishability' right-invariant equivalence relation "$\sim$" induced by B divides V^* in infinitely many equivalence classes. However, for many DCFL languages certain equivalence classes can be represented by finitely generatable words of very low complexity which is the essence of the lemma below. Let L be a DCFL language and B the accepting dpda. If $x \in V^*$ belongs to an equivalence class induced by B, then we denote this equivalence class by $[x]$. If $v_1, v_2, \ldots$ is V^* ordered lexicographically length increasing, then $\chi^x = \chi_1^x \chi_2^x \cdots$ is the *characteristic sequence* of x, such that the ith element $\chi_i^x = 1$ if $xv_i \in L$ and $\chi_i^x = 0$ otherwise. We denote $\chi_1^x \cdots \chi_n^x$ by $\chi_{1:n}^x$. We need one more notion. We say that x is *i-indistinguishable* from y, and write $x \sim_i y$, if $\chi_{1:n}^x = \chi_{1:n}^y$ with $n = |V|^i$. (I.e., $x \sim_i y$ if for all words z in V^* with $|z| \leqslant i$, either both xz, yz in L or both xz, yz not in L.)

Theorem (KC-DCFL). *Let $L \subseteq V^*$ be a DCFL, and FS_u and FS_v be finite state generators. For large enough u and v such that*

(a) *u^R and v are finitely generatable* by finite state processes FS_u and FS_v, respectively;

(b) *$v = v_n$ and χ_n^u is the mth one (or the mth zero) in χ^u considering only the words that are finitely generated by FS_v; and*

(c) $K(u) = \Omega(\log|u|)$, $K(v) = \Omega(\log|v|)$, *and* $\log\log|u| + K(m) + O(1) < K(v)$;

we have $K(\chi_{1:n}^{uv} \mid n) = O(1)$.

Proof outline. Same proof as before works. Part (1) is about the same. Part (2): The crucial part is as follows. The long and 'easy' $u' = uy$ that is finitely generated by the same process generating u is choosen such that $u' \sim_k u$, with k such that $n \leqslant |V|^k$ so that $\chi_{1:n}^{u}{}' = \chi_{1:n}^{u}$, and moreover $K(u') < \log\log|u|$. In fact,

$K(\chi^{u}_{1:n} \mid n) \leqslant |M| + \min\{K(u')$: u' is finitely generated by the same process generating $u\}$. But since $v = v_n$ is the mth one (or mth zero) in $\chi^{u'}_{1:n}$ considering only the words that are finitely generated by FS_v, we can reconstruct v using M, u' (in turn using FS_u), m and FS_v, so that $K(v) \leqslant \log\log |u| + K(m) + O(1)$. By (b) this contradicts (c). □

Clearly, the requirements (b) in the Lemma and (c) in the Theorem can be much weakened. We now give applications. All the following CFL languages were proved to be not DCFL only with great effort -- see [6].

Example 1. [Exercise 10.5 (a) in [6]]** Prove $\{w : w = w^R, w \in \{0, 1\}^*\}$ is not DCFL. Let $u = 0^n 1$ and $v = 0^n$, where $K(n) = \log n$. So they both satisfy conditions (1) and (2). Given u, uv is the first word in L with prefix u. But the first word in L with proper prefix uv would be $0^n 10^n 10^n$. So $w = 10^n$, hence $K(w) = \log n$, so L is not a DCFL by the KC-DCFL Lemma. Approximately the same proof shows that the CFL language $\{ww^R : w \in V^*\}$ and the CSL (context-sensitive) language $\{ww : w \in V^*\}$ are not DCFL languages.

Example 2. [Exercise 10.5 (b) in [6]]** Prove $\{0^n 1^m : m = n, 2n\}$ is not DCFL: Let $u = 0^n$ and $v = 1^n$, where $K(n) = \log n$. The first word in L with proper prefix uv is $0^n 1^{2n}$. So $w = 1^n$. So $K(w) = \log n$, contradicting to KC-DCFL Lemma.

Example 3. Prove $\{0^i 1^j 2^k \mid i,j,k \geqslant 0, i = j \text{ or } j = k\}$ is not DCFL. This is again easy. Let $u = 0^n$ and $v = 1^n$, where $K(n) = \log n$. So $uv \in L$. Here we apply the Corollary. Let $v' = 1^{n-1}2$. The first word in L with proper prefix uv' is $0^n 1^{n-1} 2^n$. But then $K(w) = \log n$, contradicting to the KC-DCFL Lemma (Corollary) again.

Example 4. [Pattern Matching] Prove $\{u \# v u^R w\}$ is not DCFL. Let $u = 1^n \#$ and $v = 1^n$ where $K(n) \geqslant \log n$. So uv is the first word in L with prefix u. Let $v' = 1^{n-1}0$. Then the first word in L with prefix uv' is $1^n \# 1^{n-1} 0 1^n$. So $w = 1^n$ and $K(w) \neq O(1)$, contradiction. ●

Remark. Obviously, despite of its remarkable usefulness, we do not have a proof that this KC-DCFL lemma can be used to prove that all non-DCFL CFL's are not DCFL's. Currently we can only claim that for languages the authors tested (and knew) so far, all of them can be *easily* proved using above lemma (or some variation of it). Our research in this direction has only started.

5. Recursive, Recursively Enumerable, and Beyond

It is immediately obvious how to characterize recursive languages in terms of Kolmogorov complexity. If $L \subseteq V^*$, and $V^* = \{v_1, v_2, \ldots\}$ is effectively ordered, then we define the characteristic sequence $\lambda = \lambda_1 \lambda_2 \cdots$ of L by $\lambda_i = 1$ if $v_i \in L$ and $\lambda_i = 0$ otherwise. In terms of the earlier developed terminology, if B is the automaton accepting L, then λ is the characteristic sequence associated with the equivalence class $[\epsilon]$. By definition, L is recursive if λ is a recursive sequence. It then follows trivially from the definitions:

Theorem (Recursive KC Characterisation). *A set $L \in V^*$ is recursive, iff there* *exists a constant c_L (depending on L) such that, for all n, $K(\lambda_{1:n} \mid n) < c_L$.*

L is r.e. 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] and Loveland [13].

Theorem (KC-r.e.) *(i) If L is r.e., then there is a constant c_L (depending on L), such that for all n, $K(\lambda_{1:n} \mid n) \leqslant \log n + c_L$.*
(ii)There exists an r.e. set L such that $K(\lambda_{1:n} \mid n) > \log n$.

Note that, with L as in (ii), $V^* - L$ also satisfies (i), so (i) cannot be extended to a Kolmogorov complexity characterization of r.e. sets.

Example. Fix an effective enumeration of Turing machines. Define $k = k_1 k_2 \cdots$ by $k_i = 1$ if the ith Turing machine started on a blank tape halts, and $k_i = 0$ otherwise. Let L be the language such that k is its characteristic sequence. Clearly, L is an r.e. set. We can prove that $K(k_{1:n} \mid n) > \log n$.

Example. The probability that the optimal universal Turing machine U halts on self-delimiting binary input p, randomly supplied by tosses of a perfect coin, is Ω, $0 < \Omega < 1$. Let V be a finite nonempty alphabet, and $v_1, v_2, \cdots$ an effective enumeration without repetitions of V^*. Define $L \subseteq V^*$ such that $v_i \in L$ iff $\Omega_i = 1$. It is known that $K(\Omega_{1:n} \mid n) = \Omega(n)$. Hence L, nor $V^* - L$ are r.e. It can be proved that $L \in \Delta_2 - (\Sigma_1 \cup \Pi_1)$, in the arithmetic hierarchy (i.e., L is not recursively enumerable).

6. Open Problems

(1) It is not difficult to give a KC-analogue of the *uvwxy* Pumping Lemma, as we were informed by Tao Jiang. Just like the Pumping Lemma, this will show that $\{a^n b^n c^n : n \geqslant 1\}$, $\{xx : x \in V^*\}$, $\{a^p : p \text{ is prime}\}$, and so on, are not CFL. But it fails on languages like $\{a^i b^j c^k : i \neq j \text{ or } i \neq k\}$. Clearly, this hasn't yet captured the heart of CFL. More in general, can we find a CFL-KC-Characterization?

(2) What about ambiguous CFL languages?

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

(4). Let V be a finite nonempty alphabet, w a word over V, and h a homomorphism from V^* to V^*. Then $\{h^n(w) : n \geqslant 0\}$ is called a *DOL language.* If L is a DOL language, then $g(L)$ is called a CDOL language in case g is a homomorphism from V^* to W^* such that $g(a) \neq \epsilon$ (ϵ is the empty string) for all a in V. (Cf. A. Salomaa, *Formal Languages*, Academic Press, 1973.)

Theorem. *All but finitely many words in a CDOL language are very compressible (very nonrandom).*

Proof. Let L be a CDOL language. It is easy to show that for all $x \in L$ of length n we have $K(x) \leqslant \log n + c_L$ with c_L a constant depending only on L. (Either L is finite or the number of words of length $\leqslant n$ in L satisfies

$|L^{\leqslant n}| \leqslant |V| n.)$ □

What about the remainder of the L-family of languages?

7. Addendum: New Proof of a Result of Yao and Rivest

Multihead finite and pushdown automata were studied in parallel with the field of computational complexity in the years of 1960's and 1970's. One of the major problems on the interface of the theory of automata and complexity is to determine whether additional computational resources (heads, stacks, tapes, etc.) increase the computational power of the investigated machine. A k-head deterministic (nondeterministic) one-way finite automaton, denoted as k-DFA (k-FA), is similar to a deterministic (nondeterministic) finite automaton except it has k, rather than 1, input heads. Each step, depending the current state and the k symbols read by the k heads, the machine changes its state and move some of the heads one step to the right. It stops when all heads reach the end of input.

We consider the question of whether $k+1$ heads are better than k for finite automata, and study the power of k-FA's.

Method. If the input is $abc\#a'b'c'$, and

(1) a,b,c are mutually random, i.e. $K(a\,|\,b,c) \geqslant |a| - O(\log|a|)$ and similar for b and c.

(2) $K(a'\,|\,b,c) \geqslant K(a') - O(\log|a'|)$, and similar for b' and c'.

In order to check, say, whether $a = a'$, intuitively the machine must have one head in a and another in a' simultaneously to do the matching. We will prove a lemma to make this intuition precise.

Definition. Let A be a k-FA or k-DFA. Let x and y are two blocks in the input I of A. We say that A matched x and y if on input I, there is a time A has one head in x and one head in y.

Matching Lemma. Let A accepts input $I = abc\#a'b'c'$, where $a,b,c,a',b',c' \in \Sigma^*$. Assume that A did not match b and b'. Then A accepts also input $abc\#a'b'c'$ such that

$$K(\bar{b}\,|\,a,c,a',c') \leqslant O(k^2|A|\log|I|).$$

Actually the proof will imply, and we will use, the following messier corollary to the Matching Lemma:

Corollary. Above, (1) the order is not important, for instance, b' can appear before b; (2) b' can appear as constant number of separated pieces.

Proof. Define a crossing sequence at a position p of the input to be the sequence of ID's ordered by time, where each ID contains the following information of A

$$(\textit{location of } h_1, \cdots, \textit{location of } h_k, \textit{current state})$$

at the steps some head enters position p. Each ID needs total description length at most $O(k\log|I| + |A|)$. For A, a crossing sequence contains k ID's. Let $|c.s.|$ denote the description length of a crossing sequence, then

$$|c.s.| \leqslant O(k^2|A|\log|I|).$$

Let $c.s._1$ and $c.s._2$ be the crossing sequence at the last bit of a' and the first bit of c', respectively. We search for a $\bar{b}$ as follows. For each string $\bar{b}$ of length $|b|$, simulate A on input $a\bar{b}c\#a'0^{|b'|}c'$ in the following way. Each time a head reaches the first bit of b', we check if the current status of the machine matches the description in $c.s._1$. If not, abort. If consistent, change the status of A according to $c.s._2$ and continue the simulation. If the simulation ends, then we know that A must also accept input $a\bar{b}c\#a'b'c'$ and summing up all the information we used in searching $\bar{b}$, we have $K(\bar{b} \mid a,c,a',c') \leqslant O(k^2 |A| \log |I|)$. □

The following theorem was first claimed by Rosenberg [18]. Its proof was found to be erroneous by Floyd [5]. The case $k = 2$ was proved by Ibarra and Kim [7]. Finally, the proof of the general result is due to A.C. Yao and R. Rivest [19], and C.G. Nelson [15]. We give a new proof using Kolmogorov complexity.

Theorem. $(k+1)$-head finite automata are better than k-head finite automata. More precisely, there is a language L which is accepted by a $(k+1)$-DFA but accepted by no k-FA.

Proof. Let

$$L_b = \{ w_1 \# \cdots \# w_b @ w_b \# \cdots \# w_1 \, : \, w_i \in \{0,1\}^* \}$$

as defined by Rosenberg and Yao-Rivest. Let $b = \binom{k}{2} + 1$. So L_b can be accepted by a $(k+1)$-DFA.

Assume that a k-FA M also accepts L_b. Let w be a long enough Kolmogorov random string and w be equally partitioned into $w_1 w_2 \cdots w_b$ and construct a acceptable input to M: $I = w_1 \# \cdots \# w_b @ w_b \# \cdots \# w_1$. But since $b > \binom{k}{2}$, there exists an i such that the two w_i's in I are not matched. By the matching lemma, $K(w_i \mid w - w_i) \leqslant O(k^2 |A| \log |I|) = O(\log |w|)$. But then $K(w) \leqslant |w| - |w_i| + O(\log |w|)$. We only need to make $|w_i| > O(\log |w|)$ to reach a contradiction. ●

Acknowledgements.

We thank Peter van Emde Boas, Theo Jansen, and Tao Jiang for reading and commenting on the manuscript. Tao Jiang also has suggested a KC-analogue for the *uvwxy*-lemma.

References

1. Barzdin', Y.M., "Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set," *Soviet Math. Dokl.* **9**, pp. 1251-1254 (1968).
2. Chaitin, G.J., "On the length of programs for computing finite binary sequences: statistical considerations," *J. Assoc. Comp. Mach.* **16**, pp. 145-159 (1969).
3. Chaitin, G.J., "Information-theoretic characterizations of recursive infinite strings," *Theor. Comput. Sci.* **2**, pp. 45-48 (1976).
4. Ehrenfeucht, A., R. Parikh, and G. Rozenberg, "Pumping lemmas for regular sets," *SIAM J. Computing* **10**, pp. 536-541 (1981).
5. Floyd, R., "Review 14," *Comput. Rev.* **9**, p. 280 (1968).
6. Hopcroft, J.E. and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley (1979).
7. Ibarra, O.H. and C.E. Kim, "On 3-head versus 2 head finite automata," *Acta Infomatica* **4**, pp. 193-200 (1975).
8. Kolmogorov, A.N., "Three approaches to the quantitative definition of information," *Problems in Information Transmission* **1**(1), pp. 1-7 (1965).
9. Lewis, H.R. and C.H. Papadimitriou, *Elements of the Theory of Computation*, Prentice-Hall (1981).

10. Li, M. and P.M.B. Vitányi, "Tape versus queue and stacks: The lower bounds," *Information and Computation* **78**, pp. 56-85 (1988).

11. Li, M. and P.M.B. Vitányi, "Two decades of applied Kolmogorov complexity: In memoriam A.N. Kolmogorov 1903 - 1987," pp. 80-101 in *Proc. 3rd IEEE Conference on Structure in Complexity Theory* (1988).

12. Loveland, D.W., "A variant of the Kolmogorov concept of complexity," *Information and Control* **15**, pp. 510-526 (1969).

13. Loveland, D.W., "On minimal-program complexity measures," pp. 61-65 in *Proceedings Assoc. Comp. Mach. Symposium on Theory of Computing* (1969).

14. Maass, W., "Combinatorial lower bound arguments for deterministic and nondeterministic Turing machines," *Trans. Amer. Math. Soc.* **292**, pp. 675-693 (1985).

15. Nelson, C.G., "One-way automata on bounded languages," TR14-76, Harvard University (July 1976).

16. Paul, W., "Kolmogorov's complexity and lower bounds," in *Proc. 2nd International Conference on Fundamentals of Computation Theory*, Lecture Notes in Computer Science, Vol. ??, Springer Verlag, Berlin (September 1979).

17. Paul, W.J., J.I. Seiferas, and J. Simon, "An information theoretic approach to time bounds for on-line computation," *J. Computer and System Sciences* **23**, pp. 108-126 (1981).

18. Rosenberg, A., "On multihead finite automata," *IBM J. Res. Develop.* **10**, pp. 388-394 (1966).

19. Yao, A.C.-C. and R.L. Rivest, "k+1 heads are better than k," *J. Assoc. Comput. Mach.* **25**, pp. 337-340 (1978).