Chapter 19

Kolmogorov Complexity

April 24, 2002

19.1 Introduction

Kolmogorov complexity has intellectual roots in the areas of information theory, computability theory and probability theory. Despites its remarkably simple basis, it has some striking applications in Complexity Theory. The subject was developed by the Russian mathematician Andrei N. Kolmogorov (1903–1987) as an approach to the notion of random sequences and to provide an algorithmic approach to information theory [1]. A similar theory was described by Ray J. Solomonoff (1926–) in his study of inductive inference [5, 6]. For a comprehensive treatment of this subject, including the history of this subject, we refer to the excellent book of Li and Vitányi [4].

We give some intuitions that point to Kolmogorov complexity. Consider the following binary strings:

$$\begin{aligned} x_0 &= 0000, 0000, 0000, 0000, \ldots \\ x_1 &= 1010, 1010, 1010, 1010, \ldots \\ x_2 &= 0110, 1010, 0010, 1000, \ldots \end{aligned}$$

(The commas are just a visual aid for parsing the strings). If these strings are being continued, we might ask whether the next bit can be predicted based on the first 16 bits shown. For instance, a natural guess is that the 17th bit of x_i is i for $i = 0, 1$. It is less obvious[1] what ought to be the next bit in x_2. This question leads to predictive theories about strings from examples, or more generally, the problem of inductive inference.

We can also ask for the "shortest descriptions" of strings such these. Thus x_0 might be described as "a string of zeros of lenghth 12". If x_0 is considered an infinite string, we want the shortest description of an extension of the first 12 bits. This might be "a string of all 0's". If $K(x)$ is the length of the shortest description of x (within some suitable formalism), then we can interprete $K(x)$ as the "information content" of x. One can further view those strings x where $K(x)$ is approximately $|x|$ to be "random". This makes connection with the theory of random strings. Kolmogorov complexity is essentially the study of the function $K(x)$. In the rest of this introduction, we provide some background concepts and unifying notations.

Bit Strings and Natural Numbers. *Unless otherwise noted, "strings" shall mean bit strings and "numbers" shall mean natural numbers.* We interchangeably view a bit string $x \in \{0,1\}^*$ as a number. Let ϵ denote the empty string, and $|x|$ denote the length of x. Note that xy can mean concatenation (if x, y are strings) or product (if x, y are numbers). Similarly, constants such as $0, 1, 10$ can refer to strings or numbers unless the context makes this clear. Because of such ambiguities, it is important to always distinguish between strings and numbers. To facilitate the transformations between the two domains, we introduce the bijection $\langle\cdot\rangle : \{0,1\}^* \to \mathbb{N}$ where the string $s = b_{k-1}b_{k-2}\cdots b_1 b_0$ is mapped to $n \in \mathbb{N}$ where

$$n = \sum_{i=0}^{k-1}(1 + b_i)2^i = 2^k - 1 + \sum_{i=0}^{k-1} b_i 2^i. \tag{1}$$

Thus s is just the **dyadic notation** for n, and we write $\langle s\rangle = n$. We alse define the **length** of $n = \langle s\rangle$ to be k. Setting the b_i's to all zeros or all ones in (1), we obtain

$$2^k - 1 \le n \le 2(2^k - 1)$$

[1] It is the characteristic sequence of the prime numbers (the nth bit is 1 iff n is prime).

and hence $k \leq \lg(n+1) < k+1$. This proves that the length function $\ell(n)$ is logarithmic:

$$\lg(n+1) - 1 < \ell(n) \leq \lg(n+1). \tag{2}$$

For instance, $\langle 001 \rangle = 8$ because $8 = 2 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2$. The first few values are

$$\langle \epsilon \rangle = 0, \quad \langle 0 \rangle = 1, \quad \langle 1 \rangle = 2, \quad \langle 00 \rangle = 3, \quad \langle 01 \rangle = 4, \quad \langle 10 \rangle = 5, \quad \langle 11 \rangle = 6, \quad \langle 000 \rangle = 7, \quad \langle 001 \rangle = 8, \quad \ldots.$$

We also want a notation for the inverse bijection: if $\langle s \rangle = n$, we will write $\widehat{n}$ for s. Thus $\langle \widehat{n} \rangle = n$ and $\ell(n) = |\widehat{n}|$. [The "hat" in $\widehat{n}$ is intended to suggest a connection to the angle brackets in $\langle s \rangle$.]

In Kolmogorov complexity, we need another type of encodings of numbers as bit strings. A one-one function $E : \mathbb{N} \to \{0,1\}^*$ is called a **self-limiting encoding** or **prefix-free code** of $\mathbb{N}$ if $n \neq m$ implies $E(n)$ is not a prefix of $E(m)$. It is easy to see that E cannot be a bijection. Here is the simplest prefix-free code: for $n \in \mathbb{N}$, let $E_0(n) = 1^n 0$. So $E_0(0) = 0$, $E_0(1) = 10$, $E_0(2) = 110$, etc. We will generalize E_0 to E_k for any $k \in \mathbb{N}$. Define

$$E_{k+1}(n) := E_k(\ell(n))\widehat{n}.$$

In particular, $E_1(n) = 1^{\ell(n)}0\widehat{n}$, and $E_2(n) = 1^{\ell(\ell(n))}0\widehat{\ell(n)}\widehat{n}$. So $E_0(2) = 110$ and $E_1(4) = E_0(2)\widehat{4} = 11001$. Also $E_2(4) = E_1(2)\widehat{4} = E_0(1)\widehat{\widehat{2}4} = 10101$. The reader may also verify that $E_k(0) = 0$ for all $k \geq 0$. The lengths of these encodings are given by $|E_0(n)| = 1 + n$, $|E_1(n)| = 1 + 2\ell(n)$ and $|E_2(n)| = 1 + 2\ell(\ell(n)) + \ell(n)$.

Pairing Functions. By a **pairing function**, we mean a one-one function from $\mathbb{N}^2$ to $\mathbb{N}$. The standard pairing function $\langle \cdot, \cdot \rangle'$ is defined by

$$\langle m, n \rangle' := m + \binom{m+n+1}{2}. \tag{3}$$

Thus $\langle 0,0 \rangle' = 0$, $\langle 0,1 \rangle' = 1$, $\langle 1,0 \rangle' = 2$, $\langle 0,2 \rangle' = 3$. For Kolmogorov complexity, we prefer to use a non-bijective function, $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \to \mathbb{N}$ where

$$\langle m, n \rangle = \langle E_2(m)\widehat{n} \rangle. \tag{4}$$

Note that we need to convert the string $E_2(m)\widehat{n}$ back to a number in (4). For instance, $E_2(4) = 10101$ and $E_2(1) = 0$, and so

$$\langle 4, 1 \rangle = \langle E_2(4)\widetilde{1} \rangle = \langle 101010 \rangle = 1 + 2 \cdot 2 + 4 + 2 \cdot 8 + 16 + 2 \times 32 = 105.$$

This function is a not bijection, but it has a key property which we will need:

$$\ell(\langle m, n \rangle) = \ell(n) + O(m). \tag{5}$$

So if m is fixed, we have $\ell(\langle m, n \rangle) = \ell(n) + O_m(1)$.

For $k \geq 3$, we can define the k-tupling function $\langle \cdot, \cdots, \cdot \rangle_k : \mathbb{N}^k \to \mathbb{N}$ where

$$\langle x_1, \ldots, x_k \rangle_k := \langle E_2(x_1)\langle x_2, \ldots, x_k \rangle \rangle.$$

Because we use "$\widehat{n}$" in (4), the notation $\langle m, n \rangle$ is not "selflimiting". But it suffices for our purposes. For instance, we could have defined a self-limiting version where $\langle m, n \rangle = \langle E_2(m)E_2 n \rangle$. In a certain sense, $\langle x_1, \ldots, x_k \rangle_k$ is self-limiting except for the last element of the k-tuple.

Encodings as numbers. We use angle brackets for the pairing function $\langle m, n \rangle$ as well as the dyadic notation $\langle s \rangle$. It is useful to remember this general notational convention: for any object X, we write $\langle X \rangle$ to denote a number that represents X. Here X will range over a suitable class of mathematical objects (graphs, Turing machines, etc) We will need such encodings for other classes objects later.

Computability. For any sets A and B, let $[A \to B]$ denote the set of all partial functions from A to B. Thus $[\mathbb{N} \to \mathbb{N}]$ is the set of all **number theoretic functions**. Computability theory is traditionally formulated as the study of the "computable" number theoretic functions. We take this viewpoint in Kolmogorov complexity. As in Chapter 0, we assume a fixed universal Turing machine $U^{(2)}$ such that for each $n \in \mathbb{N}$, if we place $\widehat{n}$ on the index tape of $U^{(2)}$, the resulting Turing machine computes a partial function

$$\phi_n : \mathbb{N} \to \mathbb{N}$$

assuming the dyadic notation for input and output. Thus

$$\phi_0, \phi_1, \phi_2, \ldots$$

is the **standard enumeration** of all partial recursive functions. Sometimes we need partial recursive functions of k arguments ($k \geq 3$). In this case, we will a k-tupling function that is a bijection. Furthermore, $\phi : \mathbb{N} \to \mathbb{N}$ is converted to the k-ary function $\phi^{(k)} : \mathbb{N}^k \to \mathbb{N}$ via the equation

$$\phi^{(k)}(x_1, \ldots, x_k) = \phi(\langle x_1, \ldots, x_k \rangle). \tag{6}$$

We say $\phi^{(k)}$ is **partial recursive** iff ϕ is partial recursive.

We remark that in Kolmogorov complexity, the use of simple Turing machines suffices. A further simplification comes from the fact that we may assume that the tape alphabet is binary.

Information Theory. Following Shannon, we regard a "message source" X as a random variable taking values in a finite set $S = \{x_1, \ldots, x_n\}$. We have a probability distribution $p = (p_1, \ldots, p_n)$ where $\sum_{i=1}^n p_i = 1$ and $\Pr\{X = x_i\} = p_i \geq 0$ for all $i = 1, \ldots, n$. The *entropy* $H(X)$ of the source is defined by

$$H(X) = \sum_{i=1}^{n} p_i \lg(1/p_i) = -\sum_{i=1}^{n} p_i \lg p_i.$$

For example, if $p = (1/2, 1/2)$ then $H(X) = 1$. Intuitively, on average, we need one bit to sent a message in S. If $p = (1/4, 3/4)$ the $H(X) < 0.811$ so that we need somewhat less than one bit in this case. It is not hard to show that for any n, the entropy is maximized when each $p_i = 1/n$ in which case $H(X) = \lg n$. If $n = 2^k$ then the maximum entropy is k (we need k bits to send a message in S).

Let us motivate this definition: let $I(X = x_i)$ informally measure the "information" when we receive a message $X = x_i$. First, we expect $I(X = x_i)$ to vary inversely with p_i. The message "I woke up this morning" has probability close to 1 but the information content of this message is[2] close to 0. Conversely, "I hit the first prize in the lottery" has probability close to zero, but its information content is extremely high. One choice is $I(X = x_i) = 1/p_i$. But this is not enough to ensure our next property, which is additivity of information: if X and Y are two independent message sources, the information received from X and Y to be $I(X = x_i) + I(Y = y_j)$. Both these properties are satisfied if we define

$$I(X = x_i) := \lg(1/p_i) = -\lg(p_i).$$

The base of the logarithm is not very important, but we choose $\lg = \log_2$ since most modern communication devices are binary. It follows that the above two properties hold. It is then a small step from this to the use of entropy $H(X) = \sum_{i=1}^n p_i I(X = x_i)$ as a measure of the information content of X.

Suppose Y is a second source of information that takes values y_j from some finite set. It is natural to define

$$H(XY) := -\sum_i \sum_j \Pr\{X = x_i, Y = y_j\} \lg(\Pr\{X = x_i, Y = y_j\}).$$

We also consider "relative information": how much information is conveyed by $X = x_i$ if we already know $Y = y_j$? Denote this quantity by $I(X = x_i | Y = y_j)$, or $I(x_i|y_j)$ for short. To avoid notational clutter in general, whenever we write "x_i" or "y_j" in formulas, these are understood to be short hand for "$X = x_i$" and "$Y = y_j$" (respectively). We shall define

$$I(x_i|y_j) := \lg(1/\Pr\{x_i|y_j\}).$$

It follows that $H(X|y_j) = \sum_i \Pr\{x_i|y_j\} I(x_i|y_j)$ and $H(X|Y) = \sum_j \Pr\{y_j\} H(X|y_j)$. It is not hard to show that

$$H(X|Y) \leq H(X).$$

We also have

$$H(XY) = H(X) + H(Y|X).$$

Hence, $H(XY) \leq H(X) + H(Y)$.

Note: Ziv-Lempel [2] gives a theory of compression of finite strings based on copying subsequences. See [7] for a surprising application of this.

EXERCISE

[2]The information content is not exactly zero because it tells you that I have not been sleepless the whole of last night.

Exercise 19.1.1: What is the dyadic notation for 2^n? What numbers are represented by the dyadic notation 1^n? 0^n? □

Exercise 19.1.2: Consider the prefix-free codes $E_k(n)$.
(i) What is $E_k(1)$ for all $k \geq 0$?
(ii) What is $E_k(20)$ for $k = 0, 1, 2, 3$?
(iii) What is n if $E_3(n) = 1100, 0000, 1000, 000$ (a string of length 15)?
(iv) Determine the function $f_k(n) = |E_k(n)|$ for each k. The case $k = 0, 1, 2$ has been given.
(v) Define the function $f_*(n)$ to be the limit of $f_k(n)$ as $k \to \infty$. What can you say about this function?

□

Exercise 19.1.3: Let $E : \mathbb{N} \to \{0,1\}^*$ be a self-limiting encoding. We know that E cannot be a bijection. Prove lower bounds on $|E(n)|$. □

Exercise 19.1.4: Prove that $H(X)$ is maximized when $p_i = 1/n$ for $i = 1, \ldots, n$. □

Exercise 19.1.5: Consider the function $D : \mathbb{N} \to \mathbb{N}$ where $D(n) = 1$ if $\phi_n(n) \uparrow$ and $D(n) \uparrow$ otherwise. Prove that D is not partial recursive. □

End Exercise

19.2 Kolmogorov Complexity

Let $f : \mathbb{N} \to \mathbb{N}$ be any partial function. The key concept to be investigated in Kolmogorov complexity is easily defined as follows:

$$K_f(x) := \min\{\ell(z) : f(z) = x\}.$$

If the set $\{z : f(z) = x\}$ is empty, then $K_f(x) = \infty$ by definition.

But what does $K_f(x)$ "mean"? We give three viewpoints. First, we may think of f as a **decoder function**: so $f(z) = x$ means z is the "code" and x is the "object" encoded by z. Alternatively, f is some **compiler** (or computing hardware), and z is the software or program which outputs x when executed on the hardware. Finally, f can be viewed as an **interpreter** and z is a "description" of x. The "descriptive complexity" of x is $K_f(x)$, the size of the smallest description of x.

Let us consider the third viewpoint. It may seem strange that we use natural numbers as "descriptions" for other natural numbers. But recall that natural numbers are identified with bit strings, and bit string can be used to encode any definite and representable object. For instance, let us view each bit string z as a digraph $G(z)$ (in the sense of graph theory). Then f can now be thought of as a function from $\{0,1\}^*$ to the set of (finite) graphs. Thus f can capture any an encoding of graphs. There are several well-known representation of digraphs in computer science: (i) as a list of edges, (ii) as an adjacency list or (iii) as an adjacency matrix. Note that $G(z)$ might well be one of these representations. In any case, relative to G, we could define f to capture any of the representation (i), (ii) or (iii).

Next, let

$$\mathcal{C} \subseteq [\mathbb{N} \to \mathbb{N}].$$

We say $f : \mathbb{N} \to \mathbb{N}$ is *universal* for $\mathcal{C}$ if $f \in \mathcal{C}$ and for all $g \in \mathcal{C}$, there is a constant $c = c(g)$ such that

$$K_f(x) \leq K_g(x) + c$$

for all $x \in \mathbb{N}$.

Lemma 1 *If $\mathcal{C} = [\mathbb{N} \to \mathbb{N}]$ then $\mathcal{C}$ does not have a universal function.*

Proof. Pick any $f \in \mathcal{C}$. To show that f is not universal, we construct another function $g \in \mathcal{C}$ as follows: For each $n \in \mathbb{N}$, pick some $x_n \in \mathbb{N}$ such that $K_f(x_n) \geq n$ and x_n is different from x_i for $i < n$. Such an x_n exists because we only need to pick x_n outside the finite set $\{x_0, \ldots, x_{n-1}\} \cup \{f(z) : \ell(z) < n\}$. Define $g(n) = x_n$. Then

$$K_g(x_n) = \ell(n) \leq n \leq K_f(x_n).$$

Since $K_f(z) - K_g(z) \geq n - \ell(n)$ is unbounded, we conclude that f is not universal for $\mathcal{C}$. **Q.E.D.**

We next show that the same phenomenon cannot occur if $\mathcal{C}$ is the set of partial recursive functions. However, we first generalize our definitions a little.

Conditional Kolmogorov Complexity. Let $f \in [\mathbb{N} \to \mathbb{N}]$. For any $x, y \in \mathbb{N}$, the **conditional Kolmogorov complexity of x given** y (relative to f) is defined to be

$$K_f(x|y) := \min\{\ell(z) : f(\langle y, z\rangle) = x\}. \tag{7}$$

The conditional Kolmogorov complexity function is a total function,

$$K_f : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \cup \{\infty\}.$$

If $f(\langle y, z\rangle) = x$ we call z a **f-program** for x given y. If, in addition, we have $\ell(z) = K_f(x|y)$, then z is called the **minimal f-program** for x given y. Thus, f is viewed as a programming language interpreter and $f(\langle y, z\rangle) = x$ means that "the f-program z on input y will output x".

In view of above definition, it is natural to write

$$f(z|y) := f(\langle y, z\rangle) \tag{8}$$

by inverting the order of the arguments. Thus, $f(z|y) = x$ means that z an f-program for x given y.

The **unconditional Kolmogorov complexity of** x (relative to f) is given by $K_f(x) := K_f(x|0)$. If $f(z|0) = x$ then z is an **(unconditional) f-program for** x. This new definition of $K_f(x)$ is an official replacement for the previous one, although there is practically no difference between them in our main application (when f is universal).

Since we are interested in partial computable functions, when M is a Turing machine that computes $f \in [\mathbb{N} \to \mathbb{N}]$, we may write $K_M(x)$ instead of $K_f(x)$.

We say function $\Phi \in [\mathbb{N} \to \mathbb{N}]$ is **universal** if it is partial recursive, and for all partial recursive f, there is a constant $c = c(f)$ such that for all $x, y \in \mathbb{N}$,

$$K_\Phi(x|y) \leq K_f(x|y) + c.$$

Note that the previous definition of "universality" was relative to a set $\mathcal{C}$; the new definition is similar, but with $\mathcal{C}$ equal to the set of partial recursive functions. Furthermore, we now use conditional rather than absolute Kolmogorov complexity. The starting point of Kolmogorov complexity is the following result from Kolmogorov.

THEOREM 2 (INVARIANCE) *There is a universal function Φ.*

Proof. We use the fact that there exists a partial recursive Φ with this property: for all $n, y, z \in \mathbb{N}$,

$$\Phi(\langle y, \langle n, z\rangle\rangle) = \phi_n(\langle y, z\rangle).$$

We do not care about the value of $\Phi(x)$ if x is not of the form $\langle\langle n, z\rangle, y\rangle$. We leave it as an exercise to construct a simple Turing machine to compute Φ.

Fix $n, x, y \in \mathbb{N}$. Let z be a minimal ϕ_n-program for x given y. This means $\phi_n(z|y) = x$, by (7). Therefore $\langle n, z\rangle$ is a Φ-program for x given y, since $\Phi(\langle y, \langle n, z\rangle\rangle) = \phi_n(z|y) = x$. Thus $K_\Phi(x|y) \leq \ell(\langle n, z\rangle) = \ell(z) + O_n(1)$. This proves that

$$K_\Phi(x|y) \leq K_{\phi_n}(x|y) + O_n(1)$$

. This inequality still holds if there is no ϕ_n-program for x given y, since $K_{\phi_n}(x|y) = \infty$ in this case. Thus Φ is universal. **Q.E.D.**

Based on this theorem, we can choose a fixed universal Φ and define the **conditional Kolmogorov complexity function** $K : \mathbb{N}^2 \to \mathbb{N}$ to be $K(x|y) = K_\Phi(x|y)$. Let U be the Turing machine that computes Φ. We call Φ the **reference function** and U the **reference machine** for K. A Φ-program for x (given y, etc) is simply called a **program** for x (given y, etc). The **unconditional Kolmogorov complexity function**, still denoted K, is $K : \mathbb{N} \to \mathbb{N}$ where $K_\phi(x) := K_\phi(x|0)$.

The invariance theorem tells us that if $K'(x|y)$ is defined by choosing another universal function Φ', then $|K(x|y) - K'(x|y)| \leq c$ for some constant c. So K is unique up to an additive constant.

Example. We give a typical application of the Invariant Theorem. We want to show that

$$K(\langle ww\rangle) \leq K(\langle w\rangle) + O(1) \tag{9}$$

for all strings w. Suppose z is a Φ-program for w. Consider the Turing machine M that, on input z, simulates $\Phi(z|0)$ until it halts with some output string w. Then M will duplicate the string w. If f is computed by M, we have just shown that $K_f(\langle ww\rangle) \leq K(\langle w\rangle)$. By the Invariance theorem, $K(\langle ww\rangle) \leq K_f(\langle ww\rangle) + O_f(1)$, thus proving (9).

CONVENTION: In the above example, we should like to write $K(ww)$ instead of $K(\langle ww\rangle)$, etc. In Kolmogorov complexity terminologies, we shall henceforth freely interchange numbers $m, n \in \mathbb{N}$ with their dyadic notations $\widehat{n} = v, \widehat{m} = w \in \{0,1\}^*$. We shall write $K(w)$ and $K(w|v)$ instead of the tedious but correct $K(\langle w\rangle)$ and $K(\langle w\rangle|\langle v\rangle)$. Similarly, we speak of w being a f-program for v, etc.

We next prove a logarithmic upper bound on $K(x|y)$.

THEOREM 3 *There is a constant c such that for all $x, y \in \mathbb{N}$, $K(x|y) \leq \ell(x) + c$.*

Proof. Consider the simple Turing machine M that, on input of the form $\langle y, z\rangle$ will output z. Since $\langle y, z\rangle = \langle E_2(y)\widehat{z}\rangle$, the input string is actually $E_2(y)\widehat{z}$. So M simply has to erase the prefix $E_2(y)$, and leave $\widehat{z}$ on the tape. If f is the function computed by M, then x is a program for x given y. Thus

$$K_f(x|y) = \ell(x).$$

The result then follows by the invariance theorem. **Q.E.D.**

The constant c in the preceding result depends on M but not on y. Since $K(x) = K(x|0)$, we obtain:

COROLLARY 4 *There is a constant c such that for all $x \in \mathbb{N}$, $K(x) \leq \ell(x) + c$.*

Next, we show the connection between conditional and unconditional Kolmogorov complexity.

THEOREM 5 *(i) There is a constant c such that for all $x, y \in \mathbb{N}$, $K(x|y) \leq K(x) + c$.*
(ii) For all y, there is a constant $c = c(y)$ such that for all $x \in \mathbb{N}$, $K(x) \leq K(x|y) + c$.

Proof. Assume Φ is the universal function such that $K = K_\Phi$.
(i) Let M be the STM that on input $\langle y, z\rangle$, constructs $\langle 0, z\rangle$ and simulates Φ on $\langle 0, z\rangle$. If M computes the function f, and z is a Φ-program for x, then z is a f-program for x given y. So $K_f(x|y) \leq K(x)$. The result follows from the invariance theorem.
(ii) Similarly, we construct a STM $N = N(y)$ that on input $\langle 0, z\rangle$, simulates Φ on $\langle y, z\rangle$. If N computes the function g, then $K_g(x) \leq K(x|y)$. Thus $K(x) \leq K(x|y) + O_N(1) = K(x|y) + O_y(1)$. **Q.E.D.**

Incompressibility. An important theme in Kolmogorov complexity is compressibility of strings. Because of the bijection $\langle\cdot\rangle$ between numbers and strings, we may speak of "incompressible numbers". For instance, instead of saying that the "string 0^n" is highly compressible, we will say that the number $\langle 0^n\rangle = 2^n - 1$ is highly compressible.

Let c be any constant. A number n is said to be **c-incompressible** if $K(n) \geq \ell(n) - c$. If $c = 0$, we simply say n is **incompressible**. This goes back to a motivation in studying Kolmogorov complexity: random strings. Kolmogorov intends incompressible strings to be a formalization of random strings. What is remarkable is that this notion can even be defined for strings. For, it may appear that "randomness" can only be ascribed to ω-strings.

A trivial remark is that for any $k \geq 1$ there are incompressible numbers of length k. This is because there are $\leq 2^k - 1$ minimal programs of length $< k$ but there are 2^k numbers of length k. Hence at least one number has minimial program of length $\geq k$. This is generalized to the following simple but important result:

THEOREM 6 (INCOMPRESSIBILITY) *Let $c \in \mathbb{N}$ and $A \subseteq \mathbb{N}$ be a finite set. For any $y \in \mathbb{N}$, the fraction of elements $x \in A$ such that $K(x|y) \geq \lfloor \lg |A| \rfloor - c$ is strictly more than $1 - 2^{-c}$.*

Proof. Let $|A| = m$. The number of programs (conditioned on y) of length $< \lfloor \lg(m) \rfloor - c$ is

$$\sum_{i=0}^{\lfloor \lg(m)\rfloor - c - 1} 2^i = 2^{\lfloor \lg(m)\rfloor - c} - 1 \leq m2^{-c} - 1.$$

Therefore, the number of elements $x \in A$ such that $K(x|y) \geq \lfloor \lg(m) \rfloor - c$ is $\geq m(1 - 2^{-c}) + 1$. **Q.E.D.**

When $c = 0$, this theorem says that there exists some x such that $K(x|y) \geq \lfloor \lg |A| \rfloor$. Choosing $c = 1$, we conclude that the majority of elements in A have minimal programs of length $\geq \lfloor \lg |A| \rfloor - 1$.

EXERCISE

Exercise 19.2.1: Suppose $\mathcal{C}$ contain the identity function. Prove that if f is universal for $\mathcal{C}$ then f is surjective. □

Exercise 19.2.2: Determine upper and lower bounds on the following functions.
(i) $K(n|\ell(n))$
(ii) $K(2^n)$
(iii) $K(\langle n, m\rangle|m)$
(iv) $K(\langle m, n\rangle|n)$
(v) $K(\phi_n(n))$ provided $\phi_n(n)\downarrow$. □

Exercise 19.2.3: Show some c such that:
(i) $K(\langle x, y\rangle) \leq K(x) + 2\ell(K(x)) + K(y|x) + c$.
(ii) $K(\phi_n(x)|y) \leq K(x|y) + 2\ell(k) + c$.
(iii) $K(x|\phi_n(y)) \geq K(x|y) - 2\ell(k) - c$. □

Exercise 19.2.4: Let χ be an ω-string such that $\chi[i] = 1$ iff i is prime (for $i = 1, 2, 3, \ldots$). Thus $\chi = 0110, 1010, 0010, 1000, 101$
Let $\chi[1 : n]$ be the prefix of χ of length n. Show that there is a constant $c > 0$ such that for all n, $K(\chi[1 : n]|n)\ell c$. □

Exercise 19.2.5: Construct the simple Turing machine that computes the universal function Φ described in the proof of the Invariance Theorem. Specifically, for all $n, y, z \in \mathbb{N}$,

$$\Phi(\langle y, \langle n, z\rangle\rangle) = \phi_n(\langle y, z\rangle) = \phi_n(z|y).$$

□

Exercise 19.2.6: (i) Determine the relationship between $K(x)$ and $K(\langle E_2(x)\rangle)$.
(ii) Determine $K(x|\langle E_2(x)\rangle), K(\langle E_2(x)\rangle|x)$. □

Exercise 19.2.7: Theorem 6 requires $c \in \mathbb{N}$. Show a similar result when $c \in \mathbb{R}_{\geq 0}$. □

Exercise 19.2.8: Two enumerations of the partial recursive functions $\{\phi_n\}$ and $\{\phi'_n\}$ are **recursively equivalent** if there are total recursive functions f, g such that for each n, $\phi_n = \phi'_{f(n)}$ and $\phi'_n = \phi_{g(n)}$. Prove that the invariance theorem is still true if we replace the "standard enumeration" by an equivalent one. □

End Exercise

19.3 Non-recursiveness of K

We have seen two properties of the function $K(x)$: (a) K is unbounded. This is because among strings x of length n, at least half of them satisfy $K(x) > \lg|x| - 1$. (b) K is logarithmic in the sense that $K(x) \leq \ell(x) + c$.

Another interesting property is "continuity": there is a c such that for all x, h,

$$|K(x) - K(x + h)| \leq 2\ell(h) + c. \tag{10}$$

To see this, if z is a program for x then we can construct a program z' for $x + h$ such that z' modifies the output of z by adding h to it. Moreover, $\ell(z') \leq \ell(z) + 2\ell(h) + c$. This proves (10).

We next show that K is non-recursive. Let us define $\mu : \mathbb{N} \to \mathbb{N}$ via

$$\mu(x) := \min\{K(y) : y \geq x\}.$$

We verify that

- μ is non-decreasing
- $\mu(x)$ is unbounded [since K is unbounded].
- μ is a lower bound function on $K(x)$: $\mu(x) \leq K(x)$ for all x.
- If $f(x)$ is total, non-decreasing functions that is also a lower bound on K, then $f(x) \leq \mu(x)$. In other words, μ is the unique maximal function the lower bounds on $K(x)$.

We shall show that μ is not recursive in a strong way.

THEOREM 7 *Let ϕ be partial recursive function. If ϕ is non-decreasing and unbounded then $\mu(x) < \phi(x)$ (i.o.).*

Let us first deduce a corollary.

COROLLARY 8 *If ϕ is total recursive, non-decreasing and unbounded, then ϕ is not a lower bound on K.*

We return to the theorem.

Proof. By way of contradiction, assuming $\mu(x) \geq \phi(x)$(ev.).
(1) Let A be the domain of ϕ. Hence A is r.e. and infinite. Then there is an infinite recursive subset $B \subseteq A$. Let $B = \{x_0, x_1, x_2, \ldots\}$ where $x_i < x_{i+1}$.
(2) Now define $\psi : \mathbb{N} \to \mathbb{N}$ so that $\psi(x) = \phi(x_i)$ if $x_i \leq x < x_{i+1}$. If $x < x_0$, then let $\psi(x) = 0$. Thus ψ is total recursive [pf: on input x, first determine i such that $x_i \leq x < x_{i+1}$ and then output $\phi(x_i)$.] Also, ψ is non-decreasing since ϕ is non-decreasing. Observe that

$$\psi(x) \leq \mu(x) \text{ (i.o.)}$$

since $\psi(x_i) = \phi(x_i) \leq \mu(x_i)$ (ev.i).
(3) For $a \in \mathbb{N}$, define

$$\begin{aligned} M(a) &:= \max\{x_i : \mu(x_i) \leq a\}, \\ F(a) &:= \min\{x_i : \psi(x_i) > a\}. \end{aligned}$$

Note that $M(a) < F(a)$. This is because $\mu(M(a)) \leq a < \psi(F(a)) \leq \mu(F(a))$ but μ is non-decreasing.
(4) We have $K(F(a)) > a$ since $K(F(a)) = K(x_i)$ (for some i) and $K(x_i) \geq \mu(x_i) = \psi(x_i) > a$ by definition of i.
(5) F is a recursive function (because ψ is recursive).
(6) We claim that for some $c > 0$, $K(F(a)) \leq \ell(a) + c$ for infinitely many a's. It is sufficient, by the invariance theorem, to show that $K_F(F(a)) \leq \ell(a)$ for infinitely many a's. For each $z \in \mathbb{N}$, we choose $a = \langle z, 0\rangle$. Then $\ell(a) \leq \ell(z)$. Moreover, $F(a) = F(\langle z, 0\rangle)$ implies that z is a F-program for $F(a)$ relative to 0. Thus $K_F(F(a)) \leq \ell(z) \leq \ell(a)$, as desired.
(7) Finally, items (4) and (6) gives our desired contradiction:

$$a < K(F(a)) \leq \ell(a) \text{ (i.o.)}.$$

Q.E.D.

THEOREM 9 *Let $\phi \in [\mathbb{N} \to \mathbb{N}]$ with $\texttt{domain}(\phi)$ infinite. If $\phi(x) = K(x)$ (ev. x) then ϕ is not partial recursive.*

Proof. Since $\texttt{domain}(\phi)$ is infinite and r.e., there exists a infinite recursive subset $B \subseteq \texttt{domain}(\phi)$. Define $\psi(m) := \min\{x \in B : K(x) \geq m\}$. Then the function ψ is unbounded and recursive [use fact that $K(x) = \phi(x)$ if $x \in A$ (ev.)]. Moreover

$$K(\psi(m)) \geq m \qquad \text{(by definition of } \psi) \tag{11}$$

We now claim that for some $c > 0$,

$$K(\psi(m)) \leq \ell(m) + c \text{ (i.o.)}. \tag{12}$$

By the invariance theorem, it is sufficient to show $K_\psi(\psi(m)) \leq \ell(m)$. But as in the proof of the previous theorem, if $m = \langle 0, z\rangle$ for some z (and this happens for infinitely many choices of m), then z is a ψ-program for $\phi(m)$ given 0. Thus $K_\psi(\psi(m)) \leq \ell(z) \leq \ell(m)$, proving (12). But (12) and (11) gives a contradiction. **Q.E.D.**

Length Conditioned Complexity. Theorem 7 fails if $K(x)$ is replaced by "length conditioned Kolmogorov complexity" $K'(x) := K(x|\ell(x))$. This is because $K(x|\ell(x))$, although still unbounded, will drop to a constant complexity infinitely often. For example, if $x = \langle 0^n\rangle = 2^{n+1} - 1$ then $K(x|\ell(n)) = K(2^{n+1} - 1|n) = O(1)$. Thus the corresponding lower bound function $M(x)$ is bounded, and is recursive. The complexity $K'(x)$ is interesting because we can view every string as holding two types of information: (1) its length and (2) its bit pattern. So $K'(x)$ is an attempt to measure "pure" type (2) information.

EXERCISE

Exercise 19.3.1: Let $K'(x)$ be the length conditioned Kolmogorov complexity function. Show that $K'(x) \geq K(x) - K(\ell(x))$. □

Exercise 19.3.2: Let $A \subseteq \mathbb{N}$ be any set. Let $\chi_A = b_0 b_1 b_2 \cdots$ be the ω-string such that $b_i = 1$ iff $i \in$. Write $\chi_A[i:j]$ for $b_i b_{i+1} \cdots b_j$, and $\chi_A[j]$ for $\chi_A[0:j]$.
(i) For all recursive A, there exists a constant $c = c(A)$ such that $K'(\chi_A[n]) \leq c$.
(ii) For all r.e. A, there is a constant $c = c(A)$ such that $K'(\chi_A[n]) \leq \ell(n) + c$. □

End Exercise

19.4 Some Applications

Kolmogorov Complexity has many applications, typically in lower bound proofs. For instance, in showing the existence of "random" or "hard" instances in a suitable class. Such arguments amounts to a sophisticated form of counting, and are especially amenable in the Kolmogorov Complexity framework. The advantage of such a framework is often conciseness (since the basic facts of Kolmogorov Complexity can be taken as given). Having a single framework to approach a variety of problems also a source of satisfaction.

In such applications, we will be handling general objects (Turing machines, graphs, crossing sequences, etc) as arguments to our Kolmogorov Complexity function $K(x|y)$. For instance, if G is a graph we must assume some encoding of G as a number denoted $\langle G \rangle$. Instead of writing $K(\langle G \rangle)$, we will freely write $K(G)$. In general, for any kind of object X there is an implicit encoding $\langle X \rangle$. We may need to handle a seqeuence $X_1, X_2, \ldots, X_m$ of objects, and thus need an encoding $\langle X_1, \ldots, X_m \rangle$. Instead of writing $K(\langle X \rangle | \langle X_1, \ldots, X_m \rangle)$, we simply write $K(X|X_1, \ldots, X_m)$. Furthermore, we will write $\ell(X), \ell(X_1, \ldots, X_m)$ for the length of these encodings. Another notational device is to write $(X|Y)$ (read "X given Y" instead of $\langle X, Y \rangle$. This is useful for the conditioning interpretation of arguments.

19.4.1 Crossing Sequences

We revisit the crossing sequence arguments in Chapter 2, Section 10. Throughout the following discussion, let M be a nondeterministic multitape Turing machine accepting the binary palindromes, $L_{\texttt{pal}} = \{x \in \{0,1\}^* : x = x^R\}$. Let M accept in time-space (t, s). In Chapter 2, it was shown that

$$t(n)s(n) = \Omega(n^2).$$

We now give a proof based on Kolmogorov Complexity, but assuming that M is a deterministic machine.

Recall that a storage configuration C_j is like a configuration except that the input tape contents and input head position are omitted. If a configuration is $\langle q, w_i, n_i \rangle_{i=0}^k$, then the corresponding storage configuration is just $\langle q, w_i, n_i \rangle_{i=1}^k$. If π is an accepting computation path of M on an input x of length n, and $i = 0, \ldots, n$, then an **i-crossing sequence relative to** π is $S = (C_1, \ldots, C_m)$ where C_j $(j = 1, \ldots, m)$ is the storage configuration in π when the input head of M crosses from cell i to cell $i+1$ for the $(j+1)/2$-th time (assuming odd j) or from cell $i+1$ to cell i for the $(j/2)$-th time (assuming even j). Each C_j can be represented by a string of length $O(\lg |Q| + s(3n)) = O_M(s(3n))$, where Q is the state set of M. Since $|S| = m$, we have

$$\ell(S) = O(ms(n)). \tag{13}$$

We may also assume that M always returns its input head to position 0 before accepting, and this means that we only need consider crossing sequence of even length $m = |S|$.

Lemma 10 *For any y, there exists There exists x of length n such that for all $i = \lceil n/3 \rceil, \ldots, n$, $K(x_i|y) \geq n/3 - 4\ell(n)$. Here x_i is prefix of x of length i.*

Proof. By incompressibility (Theorem 6), there exists x of length n such that $K(x|\langle M, n \rangle) \geq n$. Let U be the reference machine for K. Consider a TM N which, given $(\langle w, z \rangle | y)$, outputs $U(z|y)w$. So, if z is a U-program for x_i given y, and $x_i w = x$ then $\langle w, z \rangle$ is N-program for x given y. Since $\ell(\langle w, z \rangle) \leq \ell(z) + \ell(w) + 2\ell(\ell(w)) + 1$ and $\ell(w) = n - i$, we obtain

$$K_N(x|y) \leq K(x_i|y) + (n-i) + 2\ell(n-i) + 1 \leq K(x_i|y) + n/3 + 3\ell(n)$$

provided $\ell(n) \geq 1$. By invariance,

$$n \leq K(x|y) \leq K_N(x|y) + C \leq K(x_i|y) + 2n/3 + 4\ell(n)$$

provided $\ell(n) \geq C$. Thus $K(x_i|y) \geq n/3 - 4\ell(n)$, as claimed. Note that C depends on N and K, but not on M, n, y, x. **Q.E.D.**

We give two related definitions:
(A) A sequence S of storage configurations is called an (M, i)**-sequence** if there exists an accepting computation path π of M on some x where $|x| \geq 2i$, and S is an i-crossing sequence relative to π. Furthermore, the prefix x_i of x of length $|x_i| = i$ is called a **witness** for S.
(B) If S is any sequence of storage configurations and w a word, we say (w, S) is **compatible** iff the following Turing machine N accepts (w, S). On input $(\langle w, S\rangle|\langle M\rangle)$, N will simulate M on input w "modulo S". This means that, as long as the input head of M does read past the end of w, the simulation is normal. Let $S = (C_1, \ldots, C_m)$, m even. Immediately after the jth time ($j = 1, 2, \ldots, m/2$) when M moves its input head from position $|w| = i$ to position $i + 1$, N will check to see if the current storage configuration of M is equal to C_{2j-1}. If not, N rejects. Otherwise, N replaces the current storage configuration with C_{2j}, and continues its simulation with input head at position i. After C_m has been installed in this manner, N accepts $\langle w, S\rangle$ iff M goes on to accept its input without ever crossing to cell $i + 1$ again.

LEMMA 11 *Let S be an (M, i)-sequence.*
(i) There is a unique w of length i such that (w, S) is compatible.
(ii) There is a unique witness of length i for S.
(iii) If w is the witness for S then $K(w|M) \leq \ell(S) + 3\ell(|w|)$.

Proof.
(i) By definition of (M, i)-sequence, S has a witness w of length i. It is also clear that (w, S) is compatible. Next, for any w' of length i, we claim that if (w', S) is compatible then $w = w'$. To see this, note that since w is a witness, there is a palindrome v such that S is the $|w|$-crossing sequence relative to π, where π is the accepting computation of M on wvw^R, It follows from the compatibility of (w', S) that M also accepts $w'vw^R$. This means $w'vw^R$ is a palindrome and hence $w' = w$.
(ii) We know that (w, S) is compatible when w is a witness of S. From part (i), there is a unique u of length i such that (u, S) is compatible. We conclude that any witness of length i for S must be equal to this unique u.
(iii) Consider the Turing machine T that on input $(\langle i, S\rangle|\langle M\rangle)$ will generate each string w of length i in turn. For each w, T will check if (w, S) is compatible (using N above). If so, T outputs w. If not, T tests the next string of length i. It follows that $\langle |w|, S\rangle$ is a T-program for w given M. Hence

$$K_T(w|M) \leq \ell(|w|, S) \leq \ell(S) + 2\ell(|w|).$$

By invariance, $K(w|M) \leq \ell(S) + 2\ell(|w|) + C \leq \ell(S) + 3\ell(|w|)$, assuming $\ell(|w|) \geq C$, as desired. Note that C depends on T, and hence on N, but does not depend on M or w. **Q.E.D.**

THEOREM 12 *For all deterministic M that accepts $L_{\texttt{pal}}$ in time-space $(t(n), s(n))$, and for all $n \in \mathbb{N}$ sufficiently large, there is a constant $C > 0$ such that $t(n)s(n) \geq Cn^2$.*

Proof. By Lemma 10, there is an x of length n such that $K(x_i|M, n) \geq n/3 - 4\ell(n)$ for all $i \geq \lceil n/3\rceil$. Let S_i be the i-crossing sequence for the accepting computation path of M on input x. By Lemma 11(iii), for $i \leq n/2$, $K(x_i|M) \leq \ell(S_i) + 3\ell(n)$. Hence $\ell(S_i) \geq n/3 - 7\ell(n)$. If the length of S_i is t_i then $\ell(S_i) = Ct_i s(n))$ where C depends on M (see (13)). Summing over all $i = \lceil n/3\rceil, \ldots, \lfloor n/2\rfloor$, we obtain

$$\begin{aligned} t(n)s(n) &\geq \sum_{i=\lceil n/3\rceil}^{\lfloor n/2\rfloor} t_i s(n) \\ &\geq \sum_i C\ell(S_i) \\ &\geq C\sum_i \Big(\frac{n}{3} - 7\ell(n)\Big) \\ &= \Omega(n^2). \end{aligned}$$

Q.E.D.

19.4.2 Formal Language Theory

Li and Vitányi [3] showed that many basic results in formal language theory can be based on Kolmogorov complexity. In particular, there are analogues of the pumping lemmas. Recall that the classical pumping lemmas do not characterize regular languages. Only recently have pumping lemmas that characterize regular languages been formulated, but they are quite involved. The Kolmogorov complexity versions of the pumping lemma do characterize regular languages.

EXERCISE

Exercise 19.4.1: Obtain the lower bound on the time-space product complexity of palindromes as in the above text, but allowing nondeterministic M. □

END EXERCISE

19.5 Enumerable Real-valued Functions

We are interested in real-valued functions, and these shall be approximated by rational valued functions. In the following, let $f : \mathrm{N} \to \mathbb{R}$ and $g : \mathrm{N} \times \mathrm{N} \to \mathbb{Q}$.

The function g is said to be **recursive** if for some partial computable $\phi_n : \mathrm{N} \to \mathrm{N}$, we have that for all $m, n \in \mathrm{N}$, we have $\phi(m, n) = \langle \sigma, p, q \rangle$ for some $p, q \in \mathrm{N}$, $q \neq 0$ and $\sigma \in \{0, 1\}$, and

$$g(m, n) = (1 - 2\sigma)\frac{p}{q}.$$

We say the real-valued f is **enumerable** if there is a recursive function g as before such that $g(m, n)$ is non-decreasing in n and for all m,

$$f(m) = \lim_{n \to \infty} g(m, n).$$

Thus $g(m, n)$ approximates $f(m)$ from below. We say f is **co-enumerable** if $-f$ is enumerable (so f is approximated by some computable g from above). Finally, say f is **recursive** if there is some recursive g such that for all $m, n \in \mathrm{N}$,

$$|f(m) - g(m, n)| < 1/n.$$

LEMMA 13 *Let* $f : \mathrm{N} \to \mathbb{R}$ *and*

$$B_f := \{\langle n, \sigma, p, q \rangle : n, p, q \in \mathrm{N}, \sigma \in \{0, 1\}, (1 - 2\sigma)p/q \leq f(n)\} \in \mathrm{N}.$$

If B_f *is r.e., then* f *is enumerable.*

Proof. Construct $g(m, n)$ as follows: $g(m, 0) = (1 - 2\sigma)p/q$ where $\langle m, \sigma, p, q \rangle$ is the first value in an enumeration of B_f whose first component is m. For $n \geq 1$, let $g(m, n) = (1 - 2\sigma)p/q$ where $\langle m, \sigma, p, q \rangle$ is the first value in an enumeration of B_f whose first component is m and $(1 - 2\sigma)p/q \geq g(m, n - 1)$. **Q.E.D.**

LEMMA 14 *A real-valued function* f *is recursive if and only if it is enumerable and co-enumerable.*

Proof. ($\Rightarrow$): Let g be recursive and $|f(m) - g(m, n)| < 1/n$ for all n. Define $g_L(m, n) := \max\{g(m, i) - 1/i : i = 0, \ldots, k\}$. Note that $g_L(m, n) \leq f(m)$ and $g_L(m, n)$ is non-decreasing in n. Moreover $f(m) = \lim_{n \to \infty} g(m, n)$. We can similarly define an approximation $g_U(m, n)$ for $f(m)$ from above.
($\Leftarrow$): If g_U and g_L are upper and lower approximations for f, we define $g(m, n) := (g_U(m, n') + g_L(m, n'))/2$ where n' is the first number such that $g_U(m, n') - g_L(m, n') \leq 1/n$. **Q.E.D.**

EXERCISE

Exercise 19.5.1:
(i) Define "halting function" $HALT(m, n) = 1$ if $\phi_m(n) \downarrow$ and $HALT(m, n) = 0$ otherwise. Show that $HALT$ is not recursive.
(ii) Define the "halting set" $H := \{\langle m, n \rangle : \phi_m(n) \downarrow\}$ and show that it is r.e., but not recursive.
(iii) Define the "diagonal set" $K := \{m : \phi_m(m) \downarrow\}$ and show that it is r.e., but not recursive. □

Exercise 19.5.2: Show that the characteristic function $\chi_K : \mathrm{N} \to \{0, 1\}$ of the diagonal set K (previous exercise) is enumerable but not recursive. □

END EXERCISE

19.6 Resource-Bounded Complexity

The Kolmogorov complexity developed so far is based on static complexity complexity of Turing machines only. Traditional complexity theory is based on dynamic resources such as time and space, which are a function of actual computations. We now refine Kolmogorov complexity by considering such dynamic resource bounds. The resulting theory is considerably more intricate.

Computational Model. We now take up the off-line multitape Turing machine model of chapter 2. However, to keep the focus on number theoretic functions, $[\mathbb{N} \to \mathbb{N}]$, we shall view these machines as transducers, thus provide each with a one-way output tape. The input and output tape alphabets are assumed to be $\{0, 1\}$ but the work tapes symbols could use any alphabet. We can specify such a Turing machine T by a sequence (k, m, r, δ) where k is the number of work tapes, m is the number of work-tape symbols $\mathbb{Z}_m = \{0, 1, 2, \ldots, m-1\}$, r is the number of states $\mathbb{Z}_r = \{0, 1, \ldots, r-1\}$ and δ is a sequence of tuples from the set

$$\mathbb{Z}_r \times \mathbb{Z}_m^{k+1} \times \mathbb{Z}_r \times \mathbb{Z}_m^{k+1} \times \mathbb{Z}_3^{k+2}.$$

State 0 is the start state and state 1 is the accept state. Tape 0 is the input tape and tape $k+1$ is the output tape. A tuple

$$\langle q, a_0, \ldots, a_k, q', b_1, \ldots, b_{k+1}, d_0, \ldots, d_{k+1} \rangle \in \delta$$

of δ is interpreted as an instruction: "if T is in state q and scanning a_i's on the input and work tapes, then the next state is q' and b_i's replaces the a_i's, and the ith head moves in the direction indicated by d_i." Some of these instructions are considered illegal – for instance, those tuples that cause the output head to move left, or when q is the accept state ($q = 1$). The illegal instructions are just ignored when we use δ. The sequence (k, m, r, δ) is represented as a bit string using any reasonable convention, and when this bit string is interpreted as a number n, we say T is the nth Turing machine. We will write $\langle T \rangle = n$ in this case. Note that some numbers n may not represent any Turing machine according to our representation. In this case, by convention, n represents the **null machine** that loops on every input. This gives us a fixed enumeration

$$T_1, T_2, T_3 \ldots$$

of Turing machines where $\langle T_n \rangle = n$. We shall interprete each T_n as a deterministic Turing machine (so if more than one instruction applies to a machine configuration, we assume the first instruction in δ is applied). If the computation of T_n comes to a halt, we say it produces an output provided that final state is the accept state (state 1); otherwise it is said to produce no output. Another way to produce no output is to "loop" without halting. Assuming the 2-adic notation for input and output, let

$$\phi_n : \mathbb{N} \to \mathbb{N}$$

be the number theoretic function computed by T_n. It is not hard to see that a function is partial recursive (recall its definition as a function computed by a very simple Turing machine) iff it is equal to ϕ_n for some n. Time and space used by T_n on any input is defined as usual (we focus on "running" complexity). For instance, for any complexity function t, we say T_n runs in time t if, for any input of length n, T_n halts in $t(n)$ steps. Of course, space on the input and output tapes are not counted.

We come to the key definition: let $t, s : \mathbb{R} \to \mathbb{R} \cup \{\infty\}$ be complexity functions. Then we write

$$T_n^{t,s}(y) = x, \qquad (x, y \in \mathbb{N})$$

if T_n on input y outputs x within $t(|x|)$ steps and uses $\leq s(|x|)$ space. Otherwise we write $T_n^{t,s}(y) = \infty$. Also, let

$$K_{T_n}^{t,s}(x|y) := \min\{\ell(z) : T_n^{t,s}(\langle y, z \rangle) = x\}.$$

As usual, if $y = 0$, we may omit it to obtain the unconditional function $K_{T_n}^{t,s}(x)$. By an abuse of notation, if ϕ is computed by T_n, we shall write

$$K_\phi^{t,s}(x|y)$$

instead of $K_{T_n}^{t,s}(x|y)$. This is an abuse since time and space bounds are primarily associated with Turing machines. For the following, we shall mostly work with time bounds only and the meaning of the notations "$T_n^t(y) = x$" and "$K_\phi^t(x|y)$" should be clear.

We need a technical lemma:

LEMMA 15 *On input string w, a 1-tape Turing machine can check if $w = E_2(n)$ for some $n \in \mathbb{N}$ in $O(\ell(n))$ steps and space $\ell\ell(n)$.*

Proof. Recall that $E_2(n) = s\widetilde{n} = 1^{\ell\ell(n)}0\widetilde{\ell n}\widetilde{n}$. Using $\ell\ell(n)$ space and $|s|$ steps, we can scan w to see that it contains a prefix s of the form $s = 1^{\ell(m)}0\widetilde{m}$ for some $m \in \mathbb{N}$. If this is false, we reject at once. Otherwise, $w = ss'$ for some bit string s' and it remains to verify that $|s'| = m$. To do this, we need to view $\widetilde{m}$ (which is copied to our work tape) as a 2-adic counter. We must repeatedly decrement this counter until the zero value. For each decrement, we scan one more symbol of s', and we accept iff the zero value is reached after all of s' is scanned. We need to show that the total time for this repeated decrement takes $O(m)$ steps. This is a well-known "amortization argument" (usually applied to binary counters). The result is similar for 2-adic counter: if $\ell(m) = k$, then we subdivide the decrementing into k stages where the ith stage ($i = 1, \ldots, k$) corresponds to the counter having exactly i bits. Notice that there are at most 2^i decrement steps in the ith stage. The ith stage takes $\sum_{j=1}^{i} 2^j$ Turing machine steps. Summing over k stages, the number of steps is

$$\leq \sum_{i=1}^{k} i \cdot 2^{k+1-i} = O(2^k).$$

But $2^k = O(\ell(\ell(m)) = O(\ell(n)$. **Q.E.D.**

We have the following space-time bounded version of the Invariance theorem:

THEOREM 16 *Let s, t be complexity functions such that $s(n) \geq 1$ and $t(n) \geq n$. There exists a universal Turing machine U such that for all n, there is a $c \geq 0$ such that for all $x, y \in \mathbb{N}$,*

$$K_U^{ct\lg t,s}(x|y) \leq K_{T_n}^{t,s}(x|y) + c.$$

Proof. It is useful to first describe a simpler machine $\widetilde{U}$ which can be modified to our desired U. Let $\widetilde{U}$ be the 4-tape Turing machine that, on any input string w, checks that $\langle w\rangle = \langle y, \langle m, z\rangle\rangle$ for some $y, m, z \in \mathbb{N}$, and that T_m is a 2-tape Turing machine; we say w is well-formed in this case. If w is ill-formed, $\widetilde{U}$ will loop. Otherwise, $\widetilde{U}$ writes $\widehat{n}$ on tape 1 and $\widehat{\langle y, z\rangle}$ on tape 2, and begins to simulate T_m on input $\widehat{x}$, using tapes 3 and 4 as its worktape and tape 2 to hold its input. The contents of tape 1 is to enable U simulate T_m. Finally, $\widetilde{U}$ accepts iff T_m accepts.

Now let n be as given in the theorem. If T_n accepts within time t and space s, then the Hennie-Stearns tape reduction result (chapter 2) says that there is an $m = m(n)$ such that T_m is a 2-tape Turing machine with $\phi_m = \phi_n$ and T_m runs within time $t' = O(t \lg t)$ and space $s' = O(s)$. Suppose $T_n^{t,s}(\langle y, z\rangle) = x$ for some z. [If no such z exists, then result is trivially true.] We wish to claim that

$$\widetilde{U}^{t',s'}(\langle y, \langle m, z\rangle\rangle) = x.$$

This is not quite true, as we see.

Let us analyze the resources used by $\widetilde{U}$ on input $w = \langle y, \langle m, z\rangle\rangle$. How much time and space does $\widetilde{U}$ use before it begins the actual simulation of T_m? Note that

$$w = E_2(y)E_2(m)\widehat{z}.$$

So we can check well-formedness of w in $O(|w|)$ steps using $O_m(1)$ space, by our preceding technical lemma. We can further copy $\widehat{m}$ and $\widehat{y, z}$ (which are substrings of w) to tapes 1 and 2 while doing the checking. Similarly, assuming that our encodings of a k-tape Turing machine begins with the string $E_2(k)$, we can verify that T_m is a 2-tape Turing machine during this checking. So $\widetilde{U}$ takes $O(|w|)$ extra time steps and $O_m(1)$ space on tape 1. Unfortunately, we cannot afford the extra space $\widehat{y, z}$ on tape 2. Therefore we modify $\widetilde{U}$ to U where U does not use tape 2 at all. In U, the simulated machine T_m takes its input "$\langle y, z\rangle$" directly from the input tape of U. To do this, it needs to skip the substring $E_2(m)$ in $w = E_2(y)E_2(m)\widehat{z}$ when reading the input tape. This is easy to do, incurring $O_m(1)$ additional steps (when skipping) when simulating each step of T_m, and $O_m(1)$ additional space. Finally, U uses $O_m(1)$ steps to simulate one step of T_m. Hence the time and space used by U on inputs of length $k = |w|$ is bounded by

$$t''(k) := O_m(k + t'(k)), \quad s''(k) := O_m(s'(k) + 1).$$

This proves that $U^{t'',s''}(\langle y, \langle m, z\rangle\rangle) = x$, and so

$$K_U^{t'',s''}(x|y) \leq \ell(\langle m, z\rangle).$$

Assuming $\ell(z) = K^{t,s}_{T_m}(x|y)$, we conclude that

$$K^{t'',s''}_U(x|y) \leq K^{t,s}_{T_m}(x|y) + O_m(1).$$

Finally choose $c = c(n)$ such that $t'' \leq ct \lg t$ and $s'' \leq c \cdot s$ and $O_m(1) \leq c$. **Q.E.D.**

19.6.1 Alternative Approach.

[INCOMPLETE]

The above approach to complexity-bounded descriptions of a string x is based on the shortest program z that can "generate" the target string x. An alternative is to ask for the shortest program z that can "recognize" the target string x. Although this approach does not produce anything new in unbounded case, this is apparently different when complexity bounds are imposed.

Let $T_1, T_2, T_2, \ldots$ be the enumeration of multitape Turing machines as before. We now view each machine T_n as an acceptor, and an input x is said to be accepted by T_n if T_n finally enters the accept state (state 1). The output tape can now be ignored. Let $\psi_n : \mathbb{N} \to \{0,1\}$ be the partial function computed by T_n, where $\psi_n(x) \uparrow$ if it does not halt, and otherwise $\psi_n(x) = 1$ or 0 depending on whether T_n accepts or not; We also call ψ_n a **partial recursive predicate**. Thus we get an enumeration of all the partial recursive predicates

$$\psi_1, \psi_2, \psi_3, \ldots.$$

We define the predicate

$$T^{t,s}_n(x, y, z), \qquad (x, y, z \in \mathbb{N}) \tag{14}$$

to be be true provided, for all $y \in \mathbb{N}$, (a) if $y = x$, T_n accepts y within time $t(|y|)$ and space $s(|y|)$, and (b) if $y \neq x$, T_n does not accept y within running time $t(|y|)$ and space $s(|y|)$. Then

$$KD^{t,s}_{T_n}(x|y) := \min\{\ell(z) : T^{t,s}_n(\langle y, z\rangle) = x\}.$$

As usual, if $y = 0$, we may omit it in this notation. By an abuse of notation, if ϕ is computed by T_n, we shall write

$$K^{t,s}_\phi(x|y)$$

instead of $K^{t,s}_{T_n}(x|y)$.

Another variant definition of (14) is as follows: (a) on input y, T_n accepts iff $x = y$, and (b) if $y = x$, T_n accepts within running time $t(|x|)$ and space $s(|x|)$,

Bibliography

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

[2] A. Lempel and J. Ziv. On the complexity of finite sequences. *IEEE Trans. Inform. THeory*, IT-22:??, 1976.

[3] M. Li and P. M. B. Vitányi. A new approach to formal language theory by Kolmogorov complexity. *SIAM J. Comput.*, 24(2):398–=410, 1995.

[4] M. Li and P. M. B. Vitányi. *An introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, second edition, 1997.

[5] R. J. Solomonoff. A preliminary report on a general theory of inductive inference. Technical Report ZTB-141, Rockford Research, Cambridge, Mass., November 1960.

[6] R. J. Solomonoff. A formal theory of inductive inference, parts 1 and 2. *Information and Control*, 7:1–22, 224–254, 1964.

[7] Y. Yacobi. Fast exponentiation using data compression. *SIAM J. Comput.*, 28(2):700–703, 1998.