INFORMATION-THEORETIC CHARACTERIZATIONS OF RECURSIVE INFINITE STRINGS

Theoretical Computer Science 2 (1976), pp. 45–48

Gregory J. Chaitin
IBM Thomas J. Watson Research Center
Yorktown Heights, N.Y. 10598, USA

Abstract

Loveland and Meyer have studied necessary and sufficient conditions for an infinite binary string x to be recursive in terms of the program-size complexity relative to n of its n-bit prefixes x_n. Meyer has shown that x is recursive iff $\exists c, \forall n, K(x_n/n) \leq c$, and Loveland has shown that this is false if one merely stipulates that $K(x_n/n) \leq c$ for infinitely *many n. We strengthen Meyer's theorem. From the fact that there are few minimal-size programs for calculating a given result, we obtain a necessary and sufficient condition for x to be recursive in terms of the absolute program-size complexity of its prefixes: x is recursive iff $\exists c, \forall n, K(x_n) \leq K(n) + c$. Again Loveland's method shows that this is no longer a sufficient condition for x to be recursive if one merely stipulates that $K(x_n) \leq K(n) + c$ for infinitely many n.*

$N = \{0, 1, 2, \ldots\}$ is the set of natural numbers, $S = \{\Lambda, 0, 1, 00, 01, 10, 11, 000, \ldots\}$ is the set of strings, and X is the set of infinite strings. All strings and infinite strings are binary. The variables c, i, m and n range over N; the variables p, q, s and t range over S; and the variable x ranges over X.

$|s|$ is the length of a string s, and s_n and x_n are the prefixes of length n of s and x. x is recursive iff there is a recursive function $f : N \to S$ such that $x_n = f(n)$ for all n. $B(n)$ is the nth element of S; the function $B : N \to S$ is a recursive bijection. The quantity $|B(n)| = \lfloor \log_2(n + 1) \rfloor$ plays an important role in this paper.

A *computer* C is a partial recursive function $C : S \times S \to S$. $C(p, q)$ is the output resulting from giving C the program p and the data q. The *relative complexity* $K_C : S \times S \to N$ is defined as follows: $K_C(s/t) = \min |p| \ (C(p, t) = s)$. The *complexity* $K_C : S \to N$ is defined as follows: $K_C(s) = K_C(s/\Lambda)$. $K_C(s)$ is the length of the shortest program for calculating s on C without any data. A computer U is *universal* iff for each computer C there is a constant c such that $K_U(s/t) \leq K_C(s/t) + c$ for all s and t.

Pick a standard Gödel numbering of the partial recursive functions $C : S \times S \to S$, and define the computer U as follows. $U(0^i, q) = \Lambda$, and $U(0^i 1p, q) = C_i(p, q)$, where C_i is the ith computer. U is universal, and is our standard computer for measuring complexities. The "U" in "K_U" is henceforth omitted.

The following situation occurs in the proofs of our main theorems, Theorems 3 and 6. There is an algorithm A for enumerating a set of strings. There are certain inputs to A, e.g. n and m. $A(n, m)$ denotes the enumeration (ordered set) produced by A from the inputs n and m. And $\mathrm{ind}(s, A(n, m))$ denotes the index of s in the enumeration $A(n, m)$; $\mathrm{ind}(., A(., .)) : S \times N \times N \to N$ is a partial recursive function. A key step in the proofs of Theorems 3 and 6 is that if $s \in A(n, m)$ and one knows A, n, m, and $\mathrm{ind}(s, A(n, m))$, then one can calculate s.

Theorem 1. (a) $\exists c, \forall s, t, K(s/t) \le |s| + c$.

(b) $\forall n, t, \exists s, |s| = n$ and $K(s/t) \ge n$.

(c) There is a recursive function $f : S \times S \times N \to N$ such that $f(s, t, n) \ge f(s, t, n + 1)$ and $K(s/t) = \lim_{n\to\infty} f(s, t, n)$.

(d) $\exists c, \forall s, K(B(|s|)) - c \le K(s) \le |s| + c$.

Proof. (a) There is a computer C such that $C(p, q) = p$ for all p and q. Thus $K_C(s/t) = |s|$, and $K(s/t) \le K_C(s/t) + c = |s| + c$.

(b) There are 2^n strings s of length n, but only $2^n - 1$ programs for $U(., t)$ of length $< n$. Thus at least one s of length n needs a program of length $\ge n$.

(c) Since U is a partial recursive function, its graph $\{\langle p, q, U(p, q)\rangle\}$ is an r.e. set. Let U_n be the first n triples in a fixed recursive enumeration of the graph of U. Recall the upper bound $|s| + c$ on $K(s/t)$. Take

$$f(s, t, n) = \min\{|s| + c\} \cup \{|p| : \langle p, t, s\rangle \in U_n\}.$$

(d) Theorem 1(a) yields $K(s) = K(s/\Lambda) \le |s| + c$. And there is a computer C such that $C(p, q) = B(|U(p, q)|)$. Thus $K_C(B(|s|)) \le K(s)$ and $K(B(|s|)) \le K(s) + c$. □

Theorem 2. (a) If x is recursive, then there is a c such that $K(x_n/B(n)) \le c$ and $K(x_n) \le K(B(n)) + c$ for all n.

(b) For each c there are only finitely many x such that $\forall n$, $K(x_n/B(n)) \le c$, and each of these x is recursive (Meyer).

(c) There is a c such that nondenumerably many x have the property that $K(x_n/B(n)) \le c$ and $K(x_n) \le K(B(n)) + c$ for infinitely many n (Loveland).

Proof. (a) By definition, if x is recursive there is a recursive function $f : N \to S$ such that $f(n) = x_n$. There is a computer C such that $C(p, q) = f(B^{-1}(q))$. Thus $K_C(x_n/B(n)) = 0$ and $K(x_n/B(n)) \le c$. There is also a computer C such that $C(p, q) = f(B^{-1}(U(p, q)))$. Thus $K_C(x_n) = K(B(n))$ and $K(x_n) \le K(B(n)) + c$.

(b) See [4, pp. 525–526].

(c) See [4, pp. 515–516]. □

Theorem 3. Consider a computer D. A t-description of s is a string p such that $D(p,t) = s$. There is a recursive function $f_D : N \to N$ with the property that no string s has more than $f_D(n)$ t-descriptions of length $< K(s/t) + n$.

Proof. There are $< 2^n$ t-descriptions of length $< n$. Thus there are $< 2^{n-m}$ strings s with $\geq 2^m$ t-descriptions of length $< n$. Since the graph of D is an r.e. set, given n, m and t, one can recursively enumerate the $< 2^{n-m}$ strings having $\geq 2^m$ t-descriptions of length $< n$. Pick an algorithm $A(n,m,t)$ for doing this.

There is a computer C with the following property. Suppose s has $\geq 2^m$ t-descriptions of length $< n$. Then $s = C(0^{|p|}1pq, t)$, where $p = B(m)$ and q is the $\mathrm{ind}(s, A(n,m,t))$ th string of length $n - m$. C recovers information from this program in the following order: $|B(m)|$, m, $n - m$, n, $\mathrm{ind}(s, A(n,m,t))$, and s. Thus

$$K_C(s/t) \leq |0^{|p|}1pq| = |q| + 2|p| + 1 = n - m + 2|B(m)| + 1,$$

and $K(s/t) \leq n - m + 2|B(m)| + c$. Note that A, C and c all depend on D.

We restate this. Suppose s has $\geq 2^m$ t-descriptions of length $< K(s/t) + n$. Then $K(s/t) \leq K(s/t) + n - m + 2|B(m)| + c$. I.e., $m - 2|B(m)| - c \leq n$. Let $f_D(n)$ be two raised to the least m such that $m - 2|B(m)| - c > n$. s cannot have $\geq f_D(n)$ t-descriptions of length $< K(s/t) + n$. □

Theorem 4. There is a recursive function $f_4 : N \to N$ with the property that for no n and m are there more than $f_4(m)$ strings s of length n with $K(s) < K(B(n)) + m$.

Proof. In Theorem 3 consider only Λ-descriptions and take D to be the computer defined as follows: $D(p,q) = B(|U(p,q)|)$. It follows that there is a recursive function $f_D : N \to N$ with the property that for no n and m are there more than $f_D(m)$ programs p of length $< K(B(n)) + m$ such that $|U(p,\Lambda)| = n$. $f_4 = f_D$ is the function whose existence we wished to prove. □

Theorem 5. (a) There is an algorithm $A_5(n,m)$ that enumerates the prefixes of length n of strings s of length $2n$ such that $K(s_i) \leq |B(i)| + m$ for all $i \in [n, 2n]$.

(b) There is a recursive function $f_5 : N \to N$ with the property that $A_5(n,m)$ never has more than $f_5(m)$ elements.

Proof. (a) The existence of A_5 is an immediate consequence of the fact that $K(s)$ can be recursively approximated arbitrarily closely from above (Theorem 1(c)).

(b) By using the counting argument that established Theorem 1(b), it also follows that $\exists c, \forall n, \exists i \in [n, 2n]$ such that $K(B(i)) > |B(i)| - c$. For such i the condition that $K(s_i) \leq |B(i)| + m$ implies $K(s_i) < K(B(i)) + m + c$, which by Theorem 4 can hold for at most $f_4(m+c)$ different strings s_i of length i. Thus there are at most $f_5(m)$ elements in $A_5(n,m)$, where $f_5(m) = f_4(m+c)$. □

Theorem 6. For each c there are only finitely many x such that $\forall n, K(x_n) \leq |B(n)| + c$, and each of these x is recursive.

Proof. By hypothesis x has the property that $\forall n, K(x_n) \leq |B(n)| + c$. Thus $\forall n, x_n \in A_5(n,c)$, where A_5 is the enumeration algorithm of Theorem 5(a). There is a computer C (depending on c) such that

$$x_n = C(B(\mathrm{ind}(x_n, A_5(n,c))), B(n)) \text{ for all } n.$$

Thus

$$K_C(x_n/B(n)) \leq |B(\mathrm{ind}(x_n, A_5(n,c)))| \leq |B(f_5(c))| \text{ by Theorem 5(b).}$$

As $K_C(x_n/B(n)) \leq |B(f_5(c))|$ for all n, it follows that there is a c' such that $K(x_n/B(n)) \leq c'$ for all n. Applying Theorem 2(b) we conclude that x is recursive and there can only be finitely many such x. □

Theorem 7. x is recursive iff $\exists c, \forall n, K(x_n) \leq K(B(n)) + c$.

Proof. The "only if" is Theorem 2(a). The "if" follows from Theorem 6 and the fact that $\exists c, \forall n, K(x_n) \leq K(B(n)) + c$ implies $\exists c, \forall n, K(x_n) \leq |B(n)| + c$, which is an immediate consequence of Theorem 1(a). □

Can this information-theoretic characterization of recursive infinite strings be reformulated in terms of other definitions of program-size complexity? It is easy to see that Theorem 7 also holds for Schnorr's process complexity [5]. This is not the case for the algorithmic entropy H (see [3]). Although recursive x satisfy $\exists c, \forall n, H(x_n) \leq H(B(n)) + c$, Solovay (private communication) has announced there is a nonrecursive x that also has this property.

Theorems 6 and 7 reveal a complexity gap, because $K(B(n))$ is sometimes much smaller than $|B(n)|$.

References

[1] G. J. Chaitin, Information-theoretic aspects of the Turing degrees, Abstract 72T-E77, *AMS Notices* **19** (1972) A-601, A-602.

[2] G. J. Chaitin, There are few minimal descriptions, A necessary and sufficient condition for an infinite binary string to be recursive, (abstracts), *Recursive Function Theory Newsletter* (January 1973) 13–14.

[3] G. J. Chaitin, A theory of program size formally identical to information theory, *J. ACM* **22** (1975) 329–340.

[4] D. W. Loveland, A variant of the Kolmogorov concept of complexity, *Information and Control* **15** (1969) 510–526.

[5] C. P. Schnorr, Process complexity and effective random tests, *J. Comput. System Sci.* **7** (1973) 376–388.

Communicated by A. Meyer

Received November 1974

Revised March 1975