B. M. KLOSS

1964

SovietRxiv

Abstract

Full Text

B. M. KLOSS

ON THE DEFINITION OF THE COMPLEXITY OF ALGORITHMS

(Presented by Academician A. N. Kolmogorov, 3 II 1964)

MATHEMATICS

1. As is known, the notions of a program and of a method of programming can be defined independently of any particular refinement of the notion of an algorithm, namely: programs are natural numbers, and every method of programming is a principal enumeration of a system of partial recursive functions (for more detail see [2] and [1], § 11).

By a complexity function $K(n)$ we shall simply mean some general recursive function. For a given programming method (i.e., for a given principal enumeration $\varkappa$) one can define the so-called **complexity** of a computable function f, understanding by this the quantity $\min K(n)$, where the minimum is taken over all numbers n of the function f in the enumeration $\varkappa$.

This note clarifies the question of which general recursive functions can serve as complexity functions, so that the complexity of computable functions satisfies certain natural restrictions. Along the way, the algorithmic unsolvability of the problem of estimating the complexity of computable functions is also proved.

2. We shall call a complexity function $K(n)$ **trivial** if, for some principal enumeration, all partial recursive functions have the same complexity. We shall be interested in nontrivial complexity functions.

In what follows we shall need the following

Lemma. *Let R be an infinite general recursive set. Then there exists a principal enumeration of the system of partial recursive functions such that every function has at least one number from R.*

Proof. Since the set R is general recursive, there exists a one-to-one general recursive function $\alpha'(n)$ for which R is the set of values ([1], § 7, theorem 28). Let $\alpha''(n)$ be the function inverse to α'. The function $\alpha''(n)$ will be partial recursive ([1], § 6, theorem 12). Extend it to the following function:

$$\alpha(n) = \begin{cases} \alpha''(n), & n \in R, \\ 0, & n \in N \setminus R, \end{cases}$$

where N is the set of all natural numbers. The function $\alpha(n)$ will then be general recursive.

It is known that there exists at least one principal enumeration $\varkappa(n)$ of the system of partial recursive functions. Consider the enumeration

$$\lambda(n) = \varkappa[\alpha(n)]$$

and prove that it is principal. Indeed, let $\beta(n)$ be any computable enumeration, and let $\varphi(n)$ be a function reducing it to the enumeration $\varkappa$. Then, as is easy to see, the function $\alpha'[\varphi(n)]$ reduces the enumeration β to the enumeration λ.

It remains to prove the computability of the enumeration λ. We agree to denote the function corresponding to the number n in the enumeration λ by $\lambda_{[n]}$. Consider the function $\Psi_\lambda(n, x) = \lambda_{[n]}(x)$. This function is universal for the sys-

system of partial recursive functions. We must prove that it is itself partial recursive. But this is obvious, since it is obtained from the partial recursive function $\Psi_\chi(n, x)$ by substitution of a reducing function.

Thus, a principal numbering $\lambda(n)$ satisfying the condition of the lemma has been constructed.

Theorem 1. *In order that a general recursive function $K(n)$ be a nontrivial complexity function, it is necessary and sufficient that it assume its minimal value at only finitely many points.*

Proof. Sufficiency is obvious: if a complexity function satisfies the condition of the theorem, then it cannot be trivial, since only a finite number of computable functions will have minimal complexity.

We now prove necessity. Suppose the contrary, i.e., that the function $K(n)$ assumes its minimal value on an infinite set R of argument values. The set R is general recursive; therefore, by the lemma proved above, there exists a principal numbering which maps the set R onto the entire set of partial recursive functions. But then all partial recursive functions in this numbering have the same complexity, equal to the minimal value of the function $K(n)$.

3. An example of a nontrivial complexity function can be obtained as follows. Consider some one-to-one recursive correspondence of N with N, and assign to each $n \in N$ the length of the binary representation of the number corresponding to n. Such a function has the property that it assumes each of its values a finite number of times.

A general recursive function that assumes each of its values only a finite number of times will be called a function of **finite range**. We shall call a complexity function $K(n)$ **simple** if, for some principal numbering, the complexity of all partial recursive functions is bounded above.

Theorem 2. *In order that a general recursive function $K(n)$ not be a simple complexity function, it is necessary and sufficient that it be a function of finite range.*

Proof. Let the complexity function $K(n)$ assume each of its values a finite number of times. Then it cannot be simple. Indeed, in this case it would follow that we had numbered the system of all partial recursive functions by a finite set.

Now suppose that the complexity function $K(n)$ is not simple, but assumes some value C on an infinite set R of argument values. By virtue of the lemma proved above, one can construct a principal numbering which maps the general recursive set R onto the entire set of partial recursive functions. Then every partial recursive function has in this numbering at least one number belonging to the set R, and, consequently, its complexity does not exceed the constant C. The contradiction obtained proves our assertion.

4. **Theorem 3.** *Let $K(n)$ be a nontrivial complexity function. Then the function that assigns to each partial recursive function its complexity is not computable.*

Proof. Let the function S assign to each partial recursive function its complexity. This means that, for each number n of the given function f (with respect to the given principal numbering χ), the function S assigns the minimum of the function $K(n)$, taken over all numbers of the function f. Denoting again the function corresponding to the number n in the numbering χ by $\chi_{[n]}$, we have:

$$S(n) = \min_{\chi_{[m]}=\chi_{[n]}} K(m).$$

Suppose that the function S is general recursive. Take some value y_0 of the function S and consider the preimage R of the number y_0 under the mapping S. The set R is general recursive and includes, obviously, the set of num-

numbers of some family $\{M\}$ of partial recursive functions having complexity y_0. Since the function $K(n)$ is nontrivial, the function $S(n)$ assumes at least two values. It follows that the family $\{M\}$ does not exhaust the whole set of partial recursive functions. But membership in the family $\{M\}$ is not algorithmically recognizable (see ([1]), § 11, theorem 9); consequently, the set R cannot be general recursive. The contradiction obtained proves the theorem.

5. Thus, we have proved that there is no algorithm for computing the complexity of an arbitrary partial recursive function. But perhaps there exist "approximate" methods for computing complexity? In particular, let us consider, for example, an algorithm that computes complexity to within an additive constant. It gives rise to a function $S_a(n)$ such that

$$|S_a(n) - S(n)| < C_1, \qquad n \in N,$$

where the function $S(n)$ assigns to each function f with program n its complexity. One may consider an algorithm that computes complexity to within a multiplicative constant. It gives rise to a function $S_m(n)$ such that

$$\frac{S(n)}{S_m(n)} < C_2, \qquad n \in N.$$

Finally, let us consider an algorithm that computes complexity asymptotically. It gives rise to a function $S_\infty(n)$ satisfying the condition

$$\lim_{n\to\infty} \frac{S(n)}{S_\infty(n)} < C_3.$$

Theorem 4 proves the impossibility of all the algorithms listed and thereby gives a negative answer to the question posed above. It is curious to note that this theorem, to some extent, characterizes the "distance" between computable and noncomputable functions.

Theorem 4. Let $K(n)$ be a non-simple complexity function. Then the functions: $S_a(n)$, computing the complexity of computable functions to within an additive constant; $S_m(n)$, computing complexity to within a multiplicative constant; $S_\infty(n)$, computing complexity asymptotically, are not partial recursive.

Proof. Suppose the contrary, that the function $S_a(n)$ is partial recursive. Then, since it is everywhere defined, it will also be general recursive. Take some value y_0 of the function S_a and consider its preimage R. The set R is general recursive and consists of numbers of some family $\{M\}$ of partial recursive functions whose complexity differs from y_0 by no more than some constant C_1. Since the complexity function $K(n)$ is non-simple, the function S_a assumes an infinite set of values; therefore the family $\{M\}$ does not exhaust the whole set of partial recursive functions. The end of the proof is the same as in the preceding theorem. The proof for the function S_m is carried out analogously, and the case of asymptotic computation of complexity can be reduced to the function S_m.

Received
20 I 1964

CITED LITERATURE

1. V. A. Uspensky, *Lectures on computable functions*, Moscow, 1960. 2. V. A. Uspensky, UMN, 11, No. 4, 172 (1956).

Note: Figure translations are in progress. See original paper for figures.