ON THE COMPLEXITY OF RESOLVING ALGORITHMS

MATHEMATICS

1969

SovietRxiv

Abstract

Full Text

UDC 51.01:518.5

MATHEMATICS

M. I. KANOVICH

ON THE COMPLEXITY OF RESOLVING ALGORITHMS

(Presented by Academician P. S. Novikov, 22 XI 1968)

This paper considers questions connected with estimates of the complexity of algorithms that recognize the applicability of a given algorithm on finite sets of words.

1. We shall study normal algorithms in a certain standard extension of the alphabet A. The length of the representation of a normal algorithm $\mathfrak{A}$ will be called its **complexity** and denoted by the symbol $\mathfrak{A}$ (see ([1])).

2. By the symbol $\rho(n, \mathfrak{B}, \mathfrak{A})$, where n is a natural number and $\mathfrak{A}, \mathfrak{B}$ are normal algorithms in the standard extension of the alphabet A, we shall denote the following statement: "the algorithm $\mathfrak{B}$ is applicable to all words in the alphabet A of length not exceeding n and annuls precisely those of them to which the algorithm $\mathfrak{A}$ is applicable."

3. We shall say that a function f is a **lower estimate of the complexity of resolution** of a normal algorithm $\mathfrak{A}$, if for every natural number n, every algorithm $\mathfrak{B}$ such that $\rho(n, \mathfrak{B}, \mathfrak{A})$ has complexity not less than $f(n)$.

We shall say that a function f is an **upper estimate of the complexity of resolution** of an algorithm $\mathfrak{A}$, if for every natural number n it is false that there is no normal algorithm $\mathfrak{B}$ such that $\rho(n, \mathfrak{B}, \mathfrak{A})$ and that $\mathfrak{B} \leq f(n)$.

4. Let A be an alphabet containing at least two letters.

Theorem 1. *For every general recursive nondecreasing function f such that $\forall n f(n) \leq n$, one can specify a normal algorithm $\mathfrak{A}$ and a natural number c such that the functions g_1 and g_2, defined by the equalities*

$$g_1(n) \rightleftharpoons 1/3 f(n),$$

$$g_2(n) \rightleftharpoons f(n) + c,$$

are respectively lower and upper estimates of the complexity of resolution of the algorithm $\mathfrak{A}$.

5. The condition that the function f be majorized by a linear function in the theorem under consideration is essential, because, as Ya. M. Barzdin' and N. V. Petri have shown independently, for every algorithm $\mathfrak{A}$ one can specify a natural number c such that the function g, defined by the equality

$$g(n) \rightleftharpoons n + c,$$

is an upper estimate of the complexity of resolution of the algorithm $\mathfrak{A}$.

Thus, there is a dense "scale" of complexities of resolution of algorithms. We shall investigate the "beginning" of this "scale."

6. We shall call a word set $\mathfrak{M}$ **almost productive** if there exists an algorithm that, for every enumerable subset $\mathfrak{N}$ of the set $\mathfrak{M}$, specifies a finite list of words whose intersection with $\mathfrak{M} \setminus \mathfrak{N}$ is nonempty (cf. ([3])).

7. The following theorem has been proved:

Theorem 2. *An algorithm $\mathfrak{A}$ has an unbounded general-recursive lower bound for the complexity of solving it if and only if the complement of the domain of applicability of the algorithm $\mathfrak{A}$ is an almost productive set.*

8. As consequences of this theorem we indicate the following theorems.

Theorem 3. *Every algorithm whose domain of applicability is a creative set has an unbounded general-recursive lower bound for the complexity of its solution.*

9. **Theorem 4.** *Every algorithm whose domain of applicability is a hypersimple set has no unbounded general-recursive lower bound for the complexity of its solution.*

10. **Theorem 5.** *There exist algorithms whose domain of applicability is a simple set and which possess an unbounded general-recursive lower bound for the complexity of their solution.*

11. **Theorem 6.** *Among algorithms whose domain of applicability is a mesoic set, one can indicate both algorithms possessing an unbounded general-recursive lower bound for the complexity of their solution and algorithms for which there can be no unbounded general-recursive lower bound for the complexity of solving them.*

12. Theorem 1 also holds under a somewhat different understanding of the complexity of an algorithm (see (4)). Theorems 2–6, however, hold for a fairly general notion of complexity, namely: suppose we have some numbering of normal algorithms in a standard extension of the alphabet A, and suppose μ is some general-recursive function such that the equation $\mu(i) = n$ has a finite number of solutions for every n, and there is an algorithm indicating, for every n, the number of all solutions of this equation. Then by the **complexity** of the algorithm $\mathfrak{A}$ we shall mean the number

$\mu(k)$, where k is the number of the algorithm $\mathfrak{A}$ in the chosen numbering (see (5)).

13. The results obtained can be applied to other algorithmic problems: Post's combinatorial problem, the problem of matrix representability, problems of mathematical linguistics, etc. For example, the following theorem holds:

Theorem 7. *One can indicate such a decidable set of word systems in a two-letter alphabet that there is no general-recursive unbounded function giving a lower estimate for the complexity of algorithms deciding the compatibility of systems of bounded length, and yet there is no algorithm deciding the compatibility of systems from this set.*

The author expresses his deep gratitude to A. A. Markov for his attention and advice during the writing of this article.

Moscow State University
named after M. V. Lomonosov

Received
12 X 1968

REFERENCES

1. A. A. Markov, *Izv. AN SSSR, Ser. Matem.*, **31**, 161 (1967).
2. A. A. Markov, *Tr. Matem. Inst. im. V. A. Steklova AN SSSR*, **42** (1954).
3. J. C. Deekker, *Trans. Am. Math. Soc.*, **73**, 129 (1955).
4. A. N. Kolmogorov, *Problems of Information Transmission*, **1**, 3 (1965).
5. B. A. Trakhtenbrot, *Complexity of Algorithms and Computations*, Novosibirsk, 1967.

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