Corresponding Member of the Academy of Sciences of the USSR A. A. MARKOV

1964

SovietRxiv

Abstract

Full Text

Corresponding Member of the Academy of Sciences of the USSR A. A. MARKOV

ON NORMAL ALGORITHMS COMPUTING BOOLEAN FUNCTIONS

1. We shall call words in the alphabet 0| **Boolean vectors**; Boolean vectors of length n will be called **n-dimensional Boolean vectors**. We shall consider Boolean functions of n arguments as functions of an n-dimensional Boolean vector with admissible values 0 and |.

n-dimensional Boolean vectors may be regarded as notations for natural numbers* less than 2^n, in the binary system of numeration. It will be convenient for us to arrange these notations with the high-order digits on the right.

We agree on the following way of writing Boolean functions of n arguments by means of 2^n-dimensional Boolean vectors: the value of the Boolean function is written for the n-dimensional Boolean vector representing the number 0; on the left is appended the value of the Boolean function for the n-dimensional Boolean vector representing the number 1, and so on, until the value of the Boolean function has been written for the Boolean vector representing the number 2^n-1.

2. We shall say of a normal algorithm $\mathfrak{A}$ that it **computes** a Boolean function f of n arguments if $\mathfrak{A}(P) = f(P)$ for every n-dimensional Boolean vector P.

A normal algorithm computing a Boolean function f of n arguments, where $n > 0$, must obviously be an algorithm in some extension of the alphabet 0|. In view of the reduction theorem** every such algorithm is equivalent, relative to the alphabet 0|, to some normal algorithm in the alphabet 0|ab computing the same Boolean function. Thus, in considering normal algorithms computing Boolean functions, one may restrict oneself to algorithms in the alphabet 0|ab. We shall, however, for this same purpose use the alphabet 0|abc, which is plainly also admissible and which provides somewhat greater simplicity in the constructions. We shall denote the alphabet 0|abc by the letter Φ. Normal algorithms in the alphabet Φ will be called **Φ-algorithms**.

By the **image** of a Φ-algorithm $\mathfrak{A}$ we shall mean the word obtained from the scheme of this algorithm as follows: in each simple substitution formula the arrow is replaced by the letter x, in each final substitution formula, in place of the occurrence of the word $\rightarrow \cdot$, the letter y is substituted, at the end of each substitution formula the letter z is added, and the words so obtained are written one after another in the order in which they occur in the scheme. The length of the image of the Φ-algorithm $\mathfrak{A}$ will be called the **complexity of the algorithm** $\mathfrak{A}$ and will be denoted by the symbol $\mathfrak{A}_{\mathfrak{z}}$.

We shall say of a Boolean function f that it is m**-computable** if there exists*** a Φ-algorithm $\mathfrak{A}$ computing f and such that $\mathfrak{A} \leq m$. Here m may be any natural number. We shall say of a Boolean function f that it is m**-noncomputable** if it is not true that it is m-computable.

For fixed n and m we shall say of an algorithm $\mathfrak{A}$ that it **recognizes the m-noncomputability of Boolean functions of**

---

* We regard natural numbers here as words in the one-letter alphabet | (see ([1]) I, § 3.13).

** ([1]) III, § 7.4.2.

*** The word "exists" here and in what follows means "potentially exists" (([1]), I, § 3.5). The word "quasi-exists" means "it is false that it exists" (cf., for example, ([2]) 1.13), i.e., "the supposition of nonexistence ···leads to a contradiction."

arguments, if it is applicable to the notation of any Boolean function of n arguments and annuls the notation of a Boolean function f of n arguments if and only if f is m-noncomputable.

We shall say of a Φ-algorithm $\mathfrak{A}$ that it is a **minimal algorithm computing the Boolean function** f, if $\mathfrak{A}$ computes f and $\mathfrak{A} \preccurlyeq \mathfrak{B}$ for every Φ-algorithm $\mathfrak{B}$ computing f. We shall say of an algorithm that it **minimizes for n arguments**, if it transforms the notation of every Boolean function of n arguments into the representation of a minimal algorithm computing this function.

It is natural to construct minimizing algorithms as normal algorithms over the alphabet $0|abcxyz$. In doing so one may dispense with algorithms in the alphabet $0|abcxyzabc$. We shall denote this alphabet by the letter M. Normal algorithms in the alphabet M we shall call M-algorithms.

We shall construct **representations** of M-algorithms analogously to the way in which we constructed representations of Φ-algorithms. The length of the representation of an M-algorithm $\mathfrak{A}$ we shall call the **complexity of the algorithm** $\mathfrak{A}$ and denote by the symbol $\mathfrak{A}$.

3. Theorem 1. Every Boolean function of n arguments is (2^n+81)-computable.

Theorem 2. Whatever the natural number n, there exists a Boolean function of n arguments that is $\left[\frac{2^n}{3}\right]$-noncomputable.

Theorem 3. If a Φ-algorithm $\mathfrak{A}$ transforms every natural number n into the notation of a Boolean function of n arguments, then there exists a Φ-algorithm $\mathfrak{B}$, computing every Boolean function with notation of the form $\mathfrak{A}(n)$ $(n = 0, 1, 2, ...)$, and such that

$$\mathfrak{B} \preccurlyeq 3\mathfrak{A} + 400.$$

Corollary 1. If a Φ-algorithm $\mathfrak{A}$ transforms every natural number n into the notation of a Boolean function of n arguments, then every Boolean function with notation of the form $\mathfrak{A}(n)$ $(n = 0, 1, 2, ...)$ is $(3\mathfrak{A} + 400)$-computable.

Corollary 2. For every normal algorithm $\mathfrak{A}$ that transforms every natural number n into the notation of a Boolean function of n arguments, there can be indicated a natural number m such that every Boolean function with notation of the form $\mathfrak{A}(n)$ will turn out to be m-computable.

Corollary 3. Whatever the unbounded general recursive function φ of one argument, there is no algorithm transforming every natural number n into the notation of a $\varphi(n)$-noncomputable Boolean function of n arguments.

Theorem 4. If $m < \dfrac{2^n}{3}$ and a Φ-algorithm $\mathfrak{A}$ recognizes the m-noncomputability of Boolean functions of n arguments, then there exists a Φ-algorithm $\mathfrak{B}$, computing some m-noncomputable Boolean function of n arguments and such that

$$\mathfrak{B} \preccurlyeq 3\mathfrak{A} + 892.$$

Corollary 4. If $m < \dfrac{2^n}{3}$ and a Φ-algorithm $\mathfrak{A}$ recognizes the m-noncomputability of Boolean functions of n arguments, then

$$\mathfrak{A} > \frac{m}{3} - 298.$$

Corollary 5. If an unbounded general recursive function φ of one argument is such that for every natural number n

$$\varphi(n) < \frac{2^n}{3},$$

then there would exist no algorithm which, for every natural number n, would recognize the $\varphi(n)$-noncomputability of Boolean functions of n arguments.

Corollary 6. There exists no algorithm, applicable to every word of the form $m\square\mathfrak{F}$, where m is a natural number and $\mathfrak{F}$ is the notation of a Boolean function, that annihilates such a word if and only if the Boolean function with notation $\mathfrak{F}$ is m-noncomputable.

Theorem 5. Let $\mathcal{U}$ be a condition meaningfully imposed on Boolean functions. Then for every natural number n there quasi-exists a Φ-algorithm $\mathfrak{A}$ with the following properties:

1°. $\mathfrak{A}$ is applicable to the notation of every Boolean function of n arguments;

2°. $\mathfrak{A}$ annihilates the notation of a Boolean function f of n arguments if and only if f satisfies the condition $\mathcal{U}$;

3°.

$$\mathfrak{A}_{\ni} = 2^{2^n} + 68.$$

Corollary 7. Whatever the natural numbers n and m may be, there quasi-exists a Φ-algorithm $\mathfrak{A}$ recognizing the m-noncomputability of Boolean functions of n arguments and such that

$$\mathfrak{A}_{\ni} = 2^{2^n} + 68.$$

Theorem 6. If an M-algorithm $\mathfrak{A}$ minimizes for n arguments, then there exists a Φ-algorithm $\mathfrak{B}$ computing some

$$\left[\frac{2^n}{3}\right]\text{-noncomputable}$$

Boolean function of n arguments and such that

$$\mathfrak{B}_{\ni} \leqslant \frac{40}{11}\mathfrak{A}_{\ni} + 1561.$$

Corollary 8. Every M-algorithm minimizing for n arguments has complexity greater than

$$\frac{2^n}{11} - 430.$$

Corollary 9. There exists no algorithm minimizing for any number of arguments.

Theorem 7. Whatever the natural number n may be, there quasi-exists an M-algorithm $\mathfrak{A}$ minimizing for n arguments and such that

$$\mathfrak{A}_{\ni} \leqslant 2^{2^n}(2^n + 82) + 255.$$

4. The not very complicated proofs of Theorems 1–7 will be published elsewhere. The corollaries stated above are easily obtained on the basis of these theorems.

Let us note that in Theorems 2, 5, 7 and in Corollary 7 the prefix "quasi" cannot be omitted. For Theorem 2 this follows from Corollary 3, for Theorem 5 and Corollary 7–from Corollary 6, and for Theorem 7–from Corollary 9.

5. The author is aware that Theorems 1 and 2 have been strengthened by V. A. Kuz' min; his work is to appear shortly in *Problems of Cybernetics*. With respect to the other results, the author by no means assumes that they cannot be strengthened. Work in this direction is of great interest.

Received
7 IV 1964

REFERENCES

1. A. A. Markov, *Tr. Matem. inst. im. V. A. Steklova AN SSSR*, **42** (1954).

2. N. A. Shanin, *Tr. Matem. inst. im. V. A. Steklova AN SSSR*, **67** (1962).

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