CYBERNETICS AND CONTROL THEORY

1962

SovietRxiv

Abstract

Full Text

CYBERNETICS AND CONTROL THEORY

Yu. Ofman

ON THE ALGORITHMIC COMPLEXITY OF DISCRETE FUNCTIONS

(Presented by Academician A. N. Kolmogorov on 13 II 1962)

Let D_p be the set of numbers $1, 2, \dots, p$. In particular, $D_2 = D$ is the set consisting of the two elements 0 and 1. We shall denote by D_p^n the set of sequences $x = (x_1, x_2, \dots, x_n)$ of length n from elements of D_p. To each such sequence one can assign the numerical value

$$|x| = x_1 p^{n-1} + x_2 p^{n-2} + \cdots + x_n, \quad 0 \leq |x| < p^n.$$

We shall be interested in discrete functions $y = d(x)$ mapping D^k into D^n. The results presented below concern the estimation of the complexity of computing certain simplest functions of this kind on binary automata.

The type of binary automata suitable for us is described as follows. An automaton consists of $N \geq k + n$ "cells," numbered by natural numbers $i \leq N$. The cell with number i at time t $(t = 1, 2, \dots)$ may be in one of two states $Z_i(t) = 0$ or $Z_i(t) = 1$. The cells with numbers $i \leq k$ (input elements) are set at $t = 1$ in the states $Z_i(1) = x_i$. With each cell there is associated a function f_i, mapping D^2 into D, and two numbers r_i and s_i $(1 \leq r_i \leq N,\ 1 \leq s_i \leq N$; these numbers may coincide). The operation of the automaton proceeds in accordance with the formula

$$Z_i(t+1) = f_i[Z_{r_i}(t), Z_{s_i}(t)]. \tag{1}$$

At time T, from the cells with numbers $i > N - n$ (output elements), the result is read off as $y_j = Z_{N-j+1}(T)$. It is clear that the operation of the automaton is completely determined by: a) the numbers r_i and s_i; b) the functions f_i; c) the states $Z_i(1)$ of the cells not belonging to the input at the initial time $t = 1$; d) the running time T.

In many cases, instead of binary automata it is more convenient to use (p, q)-automata, in which each cell may be in one of p states $\in D_p$, and instead of formula (1) the formula

$$Z_i(t+1) = f_i[Z_{r_{1i}}(t), Z_{r_{2i}}(t) \dots Z_{r_{qi}}(t)], \tag{1a}$$

holds; that is, each cell is under the direct influence of q cells (distinct from one another, or coinciding). Naturally, in this case the functions f_i map D_p^q into D_p.

Fig. 1

Figure 1: Fig. 1

If a function $y = d(x)$, mapping D^k into D^n, can be computed on a (p, q)-automaton of N cells in time T, then it can be computed on a binary automaton with $N' \leq C_1 N,\ T' \leq C_2 T$, where the constants C_1 and C_2 depend only on p and q. Therefore, when studying the order of growth of N and T as some parameter grows, the distinction between binary automata and (p, q)-automata with fixed p and q is inessential.

Cascades. The definition of a cascade is inductive: the first cascade consists of the input cells; the l-th cascade consists of all cells not included in the preceding cascades but depending directly on at least one of the cells of the $(l-1)$-st cascade. It is clear that it is sufficient to consider automata in which all cells belong to cascades with numbers $l \leq T$, and, at the end of the operation at time $t = T$, the states of cells not satis-

satisfying this condition are also independent of the information introduced at the input.

Automata without feedback. By this we mean automata in which each element of cascade l depends directly only on elements belonging to the preceding cascades. In such automata the state of an element of the l-th cascade at time $t = l$ stabilizes and thereafter remains unchanged. These stabilized states of the elements uniquely determine one another, so that in essence everything reduces to computing a superposition of functions f_i. In examples 1, 2, and 3 considered below we are dealing precisely with this case; thus the estimates obtained here for N and T are estimates of the number of auxiliary variables and of the "depth of the superposition" when representing the function $y = d(x)$ by superpositions of "functions of the algebra of logic."

Algorithmic complexity of a discrete function. To each discrete binary automaton there corresponds an integral point in the positive quadrant of the plane (N, T). Among the points (N, T) corresponding to automata realizing the function $y = d(x)$, let us select those for which, to the left and below them, there are no longer any points corresponding to automata realizing d. The set Rd of the points selected by us characterizes the algorithmic complexity of the function d. It is easy to see that the points Rd can be ordered so that each subsequent one lies below and to the right of the preceding one (see Fig. 1). The set Rd ordered in this way begins with the point (N_0, T_ω) and ends with the point N_ω, T_0, where N_0 is the least number of elements in an automaton realizing d, and T is the least operating time of such an automaton. Note that the set Rd is always finite.

Fig. 1

Always

Fig. 2

Figure 2: Fig. 2

$$N_0 \geq k + n. \tag{2}$$

If, among the output symbols y_j, there is one which actually depends on each input symbol, then

$$T_0 \geq \log_2 K. \tag{3}$$

We shall call estimates (2), (3) "trivial lower estimates." In our examples 1, 2, and 3 the order of growth of N_0 and T_0 coincides with the order of growth of the trivial estimates (1), (2). At the same time, we are able to construct automata in which these smallest possible orders of growth of N and T are realized jointly, i.e., with respect to orders of growth of complexity, our three problems are solved completely.

1. An automaton for finding an element by its binary number.

Let $n = 2^k$, $k = 2^r$, and let the function $y = d_1(x)$ be defined by the condition that $y_j = 1$ for $j = |x|$; $y_j = 0$ for $j \neq |x|$.

Theorem 1. The function d_1 can be realized by a binary automaton of complexity*

$$N = 2^r + 2^{r-1+2^1} + 2^{r-2+2^2} + \ldots + 2^{1+2^{r-1}} + 2^{2^r} \asymp n;$$
$$T = r + 1 \asymp \log_2 \log_2 n.$$

* The notation $f(n) \asymp g(n)$ means that as $n \to \infty$ we have

$$0 < \underline{\lim} \frac{f(n)}{g(n)} \leq \overline{\lim} \frac{f(n)}{g(n)} < \infty.$$

The scheme of such an automaton for $r = 1$ and $r = 2$ is shown in Fig. 2. The input elements are depicted in the upper row, and the output elements in the lower row; each element depends on the two elements to which lines from above are drawn. It is easy to find the corresponding functions f_i and to continue the construction for arbitrary r.

Fig. 2

Remark. The function $y = d_2(x)$, defined by the condition that $y_j = 1$ for $j \leq |x|$, $y_j = 0$ for $j > |x|$, is realized by a $(2, 3)$-automaton with the same order of growth of the characteristics N and T.

Fig. 3

Figure 3: Fig. 3

2. An automaton for adding binary numbers. Let $k = 2m$, $n = m + 1$, and let the function $y = d_3(a_1 a_2 \ldots a_m b_1 \ldots b_m)$ be defined by

Fig. 3

the condition that $|y| = |a| + |b|$. An automaton operating according to the usual school rules has characteristics $N \asymp m$, $T \asymp m$. Nevertheless, the following holds*:

Theorem 2. The function d_3 is realized by means of a binary automaton with characteristics $N \asymp m$, $T \asymp \log_2 m$.

The usual school method of addition can be written as follows ($a_0 = b_0 = 0$, $c_{m+1} = 0$): $y_i = a_{i-1} + b_{i-1} + c_i \pmod 2$, $c_i = 1$, if $a_i + b_i + c_{i+1} \geq 2$; $c_i = 0$, if $a_i + b_i + c_{i+1} < 2$. Here c_i are the signs "kept in mind" when passing from digit to digit. If the c_i are known, then the y_i are found easily. Introduce auxiliary variables taking three values: $u_i = 0$, if $a_i + b_i = 0$; $u_i = 1$, if $a_i + b_i = 2$; $u_i = 2$, if $a_i + b_i = 1$. These variables are easily found from the values of a_i and b_i.

* From the practical point of view, the problem of reducing the number of cycles in the addition of multidigit numbers had already been studied earlier.

If u_i is equal to 0 or 1, then $c_i = u_i$. But in the case $u_i = 2$ the definition of the corresponding c_i is somewhat more difficult.

Lemma. If $m+1 = 2^r$, then there exists a $(3, 2)$-automaton with characteristics $N = 3 \cdot 2^r - r - 2 \preccurlyeq m$, $T = 2r - 1 \preccurlyeq \log_2 m$ for computing the sequence $(c_1 \ldots c_{m+1})$ from the sequence $(u_1 \ldots u_{m+1})$.

The scheme of such an automaton for $r = 16$, consisting of 7 cascades, is shown in Fig. 3. The upper row of cells is filled with the values u_i, numbered from left to right. Each of the lower cells depends on two cells from which lines come down into it from above. The corresponding function f_i is given in Table 1.

Table 1

Right argument	Left argument 0	Left argument 1	Left argument 2
0	0	1	0
1	0	1	1
2	0	1	2

The outputs are the lower cells of each vertical.

If $u_i = 0$ or 1, then this value is preserved in all cells of the corresponding vertical. If, however, $u_i = 2$, then on the corresponding vertical the two descends until,

after sufficient information has been collected about what will be carried from the lower-order digits (i.e., from the right), it is replaced by the true value c_i, equal to 0 or 1.

For $m+1$ not a power of two, one uses the nearest power of two exceeding $m+1$, and the order N and T indicated in the lemma is preserved. With the aid of the lemma, Theorem 2 is proved without difficulty.

3. Automaton for adding a large number of summands. A sequence of r binary s-digit numbers is written as a sequence x of length $k = r \cdot s$. The sum of these numbers will be no greater than $r \cdot 2^{s+1}$, and therefore can be written as a binary number y with number of digits $n = s + \log_2 r + 2$.

Theorem 3. The just-defined function $y = \bar{d}_4(x)$ is realized by a binary automaton having, as $r \to \infty$, the characteristics $N \preccurlyeq rs$, $T \preccurlyeq \log_2(r \cdot s)$.

For the proof of Theorem 3 we use the following

Lemma. There exists a $(2, 3)$-automaton with input of length $3s$ and output of length $2(s + 1)$ and operating time $T = 2$, realizing the function $y = (u_1 \dots u_{s+1}, v_1 \dots v_{s+1}) = \bar{d}_5(a_1 \dots a_s, b_1 \dots b_s, c_1 \dots c_s)$, satisfying the equality $|a| + |b| + |c| = |u| + |v|$.

This is an automaton for converting the sum of three numbers into the sum of two suitably chosen numbers. The method of constructing it is obvious if u_i, v_i are chosen in accordance with the condition $a_i + b_i + c_i = u_{i+1} + 2v_{i+1}$.

The automaton whose existence is asserted in Theorem 3 is a combination of the automata of the lemma. Combining the r summands into triples (with accuracy up to a remainder of no fewer than three summands), we replace them by approximately $2/3r$ summands, and so on, until only 2 summands remain, which are added with the aid of the automaton of Theorem 2. It is easy to compute the characteristics of the resulting composite automaton.

The results set forth were obtained in work on a broader program of research that had been outlined by A. N. Kolmogorov. Further progress in this direction has proved more difficult. Difficulties already arise in estimating the algorithmic complexity of ordinary multiplication of binary m-digit numbers.

Received
9 II 1962

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