ON THE ALGORITHMIC UNDECIDABILITY OF A-COMPLETENESS FOR THE BOUNDEDLY DETERMINATE FUNCTIONS

V. A. Buevich

A functional system $P$ is considered, whose elements are functions realized by the so-called finite automata and known as the boundedly determinate functions (B.D. functions) and whose operations are known as the operations of superposition. The system $\mathcal{F}$ of B.D. functions is called $\mathcal{A}$-complete if, for an arbitrary B.D. function and for every natural number $\tau \geq 0$, we can obtain (with the help of the operations of superposition) a B.D. function coinciding with the given one of all the words of length $\tau$ from the B.D. Functions of the system $\mathcal{F}$. The question is: does there exist an algorithm for deciding the $\mathcal{A}$-completeness of an arbitrary finite system of B.D. functions? It is shown that such an algorithm does not exist (see [4]).

We consider a functional system $P$ whose elements are functions realized by the so-called finite automata (B.D. functions) and whose operations are known as the operations of superposition. A system $\mathcal{F}$ of B.D. functions is called $\mathcal{A}$-complete if, for an arbitrary B.D. function and for every natural number $\tau \geq 0$, we can obtain, by means of the operations of superposition, a B.D. function coinciding with the given function on words of length $\tau$ from the B.D. functions of the system $\mathcal{F}$. Here we show the algorithmic undecidability of the problem of $\mathcal{A}$-completeness. We use the terminology of [1, 2].

1°. Let $E_k = \{0, 1, \ldots, k-1\}$, and let $I_k$ be the set of all the infinite sequences consisting of the elements of $E_k$. Let us denote by the element $\alpha$ of the set $I_k$ by $(\alpha(1), \alpha(2), \ldots)$. Let us also denote by $<\alpha> = (\alpha'(1), \alpha'(2), \ldots)$ and $<\alpha> = (\alpha''(1), \alpha''(2), \ldots)$ the elements $\alpha'$ and $\alpha''$ of the set $I_k$ such that for every $t = 1, 2, \ldots$, $\alpha'(t) = \alpha(2t-1)$ and $\alpha''(t) = \alpha(2t)$. The variables taking values in the set $I_k$ will be denoted by the symbols $x, y,$ and $z$ with indices. We shall represent every such variable, say $x$, in the form $x = (x(1), x(2), \ldots)$. By the boundedly determinate function (B.D. function) $T(x_1, \ldots, x_n) = y$ we understand a function from the set $I_k \times I_k \times \ldots \times I_k$ into $I_k$ defined by the following recursive equations:

$$
\begin{align*}
q(1) &= q_0, \\
q(t+1) &= \Psi(x_1(t), \ldots, x_n(t), q(t)), \\
y(t) &= \Phi(x_1(t), \ldots, x_n(t), q(t)),
\end{align*}
$$

where the function $\Psi$ takes values in the finite alphabet $Q = \{q_0, q_1, \ldots, q_s\}$, whose elements are called the states of the B.D. function $T$, and $q_0$ is called the initial state. Equations (1) are usually called, the canonical equations of the B.D. function.

The B.D. function $T(x_1, \ldots, x_n)$ depends essentially on the variable $x_1$, if there exist two aggregates $(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n)$ and $(a_1, \ldots, a_{i-1}, a'_i, a_{i+1}, \ldots, a_n)$, where $a_i \in I_k$, $i = 1, 2, \ldots, n$, $a'_i \in I_k$ such that $T(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n) \neq T(a_1, \ldots, a_{i-1}, a'_i, a_{i+1}, \ldots, a_n)$. In this case the variable $x_1$ is called essential. The variables which are not essential are called fictitious.

Let $\mathfrak{M} \subseteq P$. Let us introduce, in a recursive manner, the notion of a superposition over $\mathfrak{M}$:

1. Every B.D. function of $\mathfrak{M}$ is a superposition over $\mathfrak{M}$.

2. If \( F(x_1, \ldots, x_n) \) is a superposition over \( \mathcal{M} \), then every function obtained from \( F \) by the addition or deletion of an arbitrary number of fictitious variables is also a superposition.

3. If \( F(x_1, \ldots, x_n) \) and if every function \( G_i(x_1, \ldots, x_m) \ldots G_n(x_1, \ldots, x_m) \) is either a superposition over \( \mathcal{M} \) or for some \( i \), \( G_i(x_1, \ldots, x_m) = x_j \) where \( i = 1, 2, \ldots, n \), \( j = 1, 2, \ldots, m \), then the composite function \( H(x_1, \ldots, x_m) = F(G_i(x_1, \ldots, x_m), \ldots, G_n(x_1, \ldots, x_m)) \) is a superposition over \( \mathcal{M} \).

We can show that a superposition of B. D. functions is also a B. D. function. Let us denote the set of all superpositions over \( \mathcal{M} \) by \( \mathcal{M}_1 \), and let us call it the completion of the set \( \mathcal{M} \). The set \( \mathcal{M} \) is called complete if \( \mathcal{M}_1 = \mathcal{M} \). The functions \( T_1(x_1, \ldots, x_n) \) and \( T_2(x_1, \ldots, x_n) \) are called \( \tau \)-equivalent \( (T_1 \sim T_2) \) if for an arbitrary aggregate \( (a_1, \ldots, a_n) \), where \( a_i \in I_k \) \( (1 \leq i \leq n) \), the corresponding first \( \tau \)-digits of the sequences \( T_1(a_1, \ldots, a_n) \) and \( T_2(a_1, \ldots, a_n) \) coincide. The sets \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \) are \( A \)-equivalent \( (\mathcal{M}_1 \sim \mathcal{M}_2) \), if, for an arbitrary natural number \( \tau > 0 \) and an arbitrary B. D. function belonging to the completion of one of the sets, we can find a B. D. function in the completion of the other set, which is \( \tau \)-equivalent to the given one. The set \( \mathcal{M} \) is called A-complete if its completion is \( A \)-equivalent to \( \mathcal{M} \). The B. D. function \( T(x_1, \ldots, x_n) \) is called A-universal if the set consisting of only this function is A-complete.

It is obvious that every complete set is A-complete. The converse is, in general, not true. Moreover, not every finite A-complete set is complete. For example, the set consisting of the single B. D. function \( T_0(x_1, x_2) \), which has been constructed in [3], is A-complete. This B. D. function is obviously A-universal.

2. Let us recall some facts from the theory of algorithms [2]. Let \( \mathcal{U} \) be a finite alphabet, \( E_0 \) be a fictitious word, called the primitive, in this alphabet, and \( \mathcal{B} = \{ (P_0; Q_0), \ldots, (P_n; Q_n) \} \) be a system of pairs of words in this alphabet. Each such system of pairs of words defines the normal Post calculus \( P(\mathcal{B}) \) with the primitive word \( L_0 \). Every law of direct deducibility of the calculus \( P(\mathcal{B}) \) transforms any word having the form \( P_1R \), where \( R \) is an arbitrary word in the alphabet \( \mathcal{U} \) into the word \( RQ \). We say that a word \( C \) is deducible in the calculus \( P(\mathcal{B}) \), if \( C = L_0 \) or if there exists a sequence of words \( B_1, \ldots, B_n \) such that \( B_1 = L_0 \), \( B_n = C \) and any word \( B_j \) is obtained from the word \( B_{j-1} \) \( (0 < i \leq n) \) by applying one of the laws of direct deducibility of the calculus \( P(\mathcal{B}) \). In [2] is given a concrete example of normal Post calculus, for which the problem of deducibility turns out to be algorithmically undecidable. Let this calculus be designated as \( C_4 \). The alphabet \( \mathcal{U} \) of the calculus \( E_4 \) consists of 14 letters, and this calculus has 88 laws of direct deducibility. As the concrete form of the calculus is not essential in what follows, let us assume that the alphabet \( \mathcal{U} = \{ 3, 4, \ldots, 15, 16 \} \), the primitive word \( L_0 = p_1 \ldots p_q \), where \( p_q \in \mathcal{U} \), and the 88 pairs of words corresponding to the laws of direct deducibility of the calculus \( E_4 \) can be written in the following form:

\[
\begin{align*}
(P_0; Q_0) & \quad \text{are} \quad n_0 \ldots n_{0r} \ldots m_0 \ldots m_{sa} \\
(P_1; Q_1) & \quad \text{are} \quad n_1 \ldots n_{1v} \ldots m_1 \ldots m_{sa} \\
\ldots & \quad \ldots \\
(P_{87}; Q_{87}) & \quad \text{are} \quad n_{87} \ldots n_{87l} \ldots m_{87} \ldots m_{87l},
\end{align*}
\]

where \( n_{ij} \) \( (0 \leq i \leq 87, 1 \leq j \leq r) \) and \( m_{ij} \) \( (0 \leq k \leq 87, 1 \leq l \leq s_k) \) are letters of the alphabet \( \mathcal{U} \).

3. As noted above, there exist finite A-complete systems of B. D. functions. Then the question arises as to whether there exists an algorithm for deciding the A-completeness of an arbitrary finite system of B. D. functions. It is shown that such an algorithm does not exist. This fact can be established by reducing the problem of deciding A-completeness to the problem of deducibility in the calculus \( E_4 \). Namely, to an arbitrary word \( L \) in the alphabet \( \mathcal{U} \) we associate a corresponding system \( S(L) \) consisting of 4 B. D. functions, and we show that the system \( S(L) \) is A-complete if and only if the word \( L \) is not deducible in the calculus \( E_4 \). Moreover, we can reduce the problem of deciding A-universality of an arbitrary B. D. function to the problem of deducibility in the calculus \( E_4 \) and also show that this problem is algorithmically undecidable.

Let us consider \( E_{88} = \{ 0, 1, \ldots, 87 \} \) and assume that the input and the output variables of the B. D. functions take values in \( E_{88} \). Let \( L' = l_1 \ldots l_\alpha \) be a word in the alphabet \( \mathcal{U} \). We shall call every word \( L'' = l_\sigma \ldots l_\alpha \) as the beginning of the word \( L' \) if \( l_\sigma = l_1 \ldots l_\sigma \) where \( 1 \leq \sigma \leq \alpha \). To every word \( L = l_1 \ldots l_\alpha \) in the alphabet \( \mathcal{U} \) let us associate the corresponding system of B. D. functions.

\[
S(L) = (T_{1h}(x), T_{2b}(x), T_{3b}(x_1, x_2), T_{4b}(x_1, x_2)).
\]