AN ITERATIVE DESCRIPTION OF THE CLASS $\mathcal{E}'$ OF GRZEGORCZYK'S HIERARCHY

A. P. Bel'tyukov

A class of abstract computing machines is described. It is proved that the class of all functions computable by these machines coincides with the class $\mathcal{E}'$ of Grzegorczyk's hierarchy.

The classes $\mathcal{E}^n$ of the Grzegorczyk hierarchy for $n \geq 2$ may be characterized as the classes of functions computed on Turing machines under a certain bound on the length of the computation tape used (see, e.g., [1]). Such a representation for the classes $\mathcal{E}^n$ and is unknown. In our paper, a certain type of abstract computing machine is described and it is proved that $\mathcal{E}'$ is the class of all functions computable on these machines. A similar result, that $\mathcal{E}'$ consists of all functions computable on certain machines, was announced in [2].

The original definition of the classes $\mathcal{E}^n$ was given in [3]. Below a definition of $\mathcal{E}'$ is given which is easily seen equivalent to the original one. The class $\mathcal{E}'$ is the class of all functions (defined on the natural, i.e., the nonnegative, integers) obtained from the functions $Z, I^n, S, +$ by composition and bounded primitive recursion, where the functions $Z, I^n, S$ are defined by the equations $Z(x)=0$, $I^n(x_1, \ldots, x_n)=x_n$, $S(x)=x+1$ for all $x, x_1, \ldots, x_n$.

Below we will use the following abbreviations. The sequence $x_1, \ldots, x_k$ is denoted $x_{1:k}$ and $\bar{x}$; the sequence $t_1, \ldots, t_i$ is denoted $t_{1:i}$ and $\bar{t}$; the sequence $t_1, \ldots, t_i$ is denoted $V$ (for $i \leq \tau$); the number $\lambda \{ \Sigma_{t \leq \lambda} x_i + \Sigma \} \in \mathcal{V}$ is denoted $\mathcal{V}$; $\mathcal{U}$ denotes the closure of the set of functions

$$\left\{ Z, S, +, \lambda x y z \Sigma \{ y(z) \} \cup \cup_{n \leq 1} \bigcup_{n \leq 1} \{ I^n \} \right\}$$

under composition. It is not hard to see that $\mathcal{U}$ is a subset of $\mathcal{E}'$.

1. Definitions and Results

1. The machines considered below have a memory consisting of a finite set of registers but no tape, such as a Turing machine has. In any register, at each moment of time there is a natural number. A special program controls the changes of the memory. Important elements of these programs are the names of registers, which are certain symbols which correspond to the registers. The registers have the following four forms:

1A. Input registers. These registers have the names $X_1, X_2, \ldots$.

1B. Timers, having the names $T_1, T_2, \ldots$.

1C. A working register, named $S$.

1D. A null register, named $0$.
2. A transfer instruction is a word of the form \((Q=R|k|l)\), where \(Q\) and \(R\) are names of registers, \(k\) and \(l\) are integers (written in some system of notation). A copy instruction is a word of the form \(S:=Q\), where \(Q\) is a register name. An instruction is either a transfer instruction or a copy instruction.

3. We will say that the instruction with number \(k\) in the list \(\langle J_0; J_1; \ldots; J_{n-1} \rangle\) may transfer to addresses \(i\) and \(j\) if \(J_k\) has the form \((Q=R|i-k|j-k)\).

A timer program is a nonempty list of instructions \(\langle J_0; J_1; \ldots; J_{n-1} \rangle\), having the following properties.

3A. No instruction in the list may transfer to a negative address or to an address greater than \(n-1\).

3B. The instruction \(J_{n-1}\) is a transfer instruction.

3C. There is no list of natural numbers \(\langle k_1, k_2, \ldots, k_n \rangle\) such that the instruction with number \(k_1\) may transfer to address \(k_2\), and the instruction with number \(k_2\) may transfer to address \(k_3\), etc., and the instruction with number \(k_n\) may transfer to address \(k_1\); i.e., there are no cycles of only transfer instructions.

The number of timers \(\Sigma(P)\) used by the program \(P\), is \(\max \{l|T_l \subset P \land \forall i \neq l: T_i \neq l\}\), where \(A \subset P\) means that word \(A\) is a subword of one of the words in the list \(P\). The arity \(\chi(P)\) (the number of input registers) of the program \(P\) is \(\{i|\chi_i \subset P \land \forall i \neq l: \chi_i \neq l\}\). Thus if \(P\) contains names of timers, \(\Sigma(P)\) is the largest index of the timer names; otherwise, \(\Sigma(P) = 1\). Analogously, if \(P\) contains the names of input registers, then \(\chi(P)\) is the largest index of these names; otherwise \(\chi(P) = 1\).

4. Suppose some timer program is given consisting of \(\tau\) instructions, having \(\tau\) timers and having arity \(\xi\). The configurations are lists of the form

\[ \langle v, x_{i:1}^{\xi}, t_{c:1}, a, k \rangle, \]

where \(v, x_{i:1}^{\xi}, t_{c:1}, a, k\) are natural numbers satisfying the following inequalities:

\[ v > 1, x_i, t_i, a, k < v, t_\xi < v, \ldots, t_i < v, \ldots, t_\xi < v, a < v, k < n-1. \]

We will call \(v\) the size register, the number \(x_i\) is the value (i.e., the content) of the input register \(X_i\), the number \(t_i\) is the value of the timer \(T_i\), the number \(a\) is the value of the working register \(S\), and the number \(k\) is the number of the instruction being executed. The value of the register \(0\) is always the number \(0\).

The time of configuration (1) is, by definition, the number \(t_\xi^\tau(\xi+1)^{\xi-1} + t_{\xi-1}^\tau(\xi+1)^{\xi-2} + \ldots + t_i^\tau\).

The execution of the instruction \((Q=R|l|i)\) in configuration (1) consists in changing the number of instruction being executed to \(k+i\), if the value of \(Q\) is equal to the value of \(R\), and to \(k+j\) otherwise.

In the first stage of the execution of the instruction \(S:=Q\) in configuration (1), the timers change their values so that the time of the configuration is extended by 1 and their new values do not exceed \(v\). If this is impossible, all timers change their values to \(0\). It is not difficult to see that the new values of the timers are uniquely defined. In the second stage,