Computer Science Journal of Moldova, vol.**6**, no.1(16), 1998

Some small self-describing Turing machines

M. Margenstern Yu. Rogozhin

Abstract

Several small self-describing Turing machines are constructed. The number of instructions varies from 275 to 206, depending on the chosen encoding principles. The previous Thatcher's result (2532 instructions of self-describing machine in Wang's format) is essentially improved.

1 Introduction

This paper was motivated by von Newmann's works [6, 7] on self-reproducing machines, and Lee's [4] and Thatcher's [8] works to pose the question: is it possible for Turing machine to print on its tape a complete description of its "internal structure"? The answer is "yes", i.e. these machines exist and are called *self-describing Turing machines*. Having adopted the convention of identifying "internal structure" with program of Turing machine, the problem becomes that of finding *the simplest* program, the program with the minimal number of instructions, which can print out its own description. But because description is not well-defined term, there are several approaches which one could take to make the original question precise [4, 8, 9]. Each of these approaches is based on the identification of "description" with "Gödel numbering", that is, a description is obtained through an effective correspondence (encoding) between programs and natural numbers or sequences of natural numbers.

For any instruction of a given program several instructions are required to write the encoding (or description) of that instruction. Thus, any brute-force attempt at obtaining a self-describing machine is hopeless, as the program must become indefinitely long.

Standard methods for overcoming this apparent paradox are known (see Lee [4], he used universal Turing machine for that and his method takes about 166000 instructions, and original decision of Thatcher [8] without universal Turing machine, his method takes 2532 instructions in Wang's format (see [11]). We describe the Thatcher's method and its improvement below.

The results of this paper are presented at CAW'97, Cellular Automata Workshop, Gargnano (Italy), September, 1997 [5].

This paper contents some results about self-describing Turing machines obtained by authors in 1997. The full variant of the paper about self-describeness will be ready later.

2 The existence of a self-describing Turing machine

The self-describeness of a Turing machine means that the machine gives its own description. In order to focus on this property, it is usually assumed that the initial configuration is the blank tape. Recall, at this point, that we deal only with deterministic Turing machines (TM) with a single head and a single one-dimensional tape, infinite in both directions.

The instructions of TM are: $< q_i, a_j, a_l, \delta, q_k >$ or simple $q_i a_j a_l \delta q_k$, where $i, k \in \{1, 2, \ldots, n\}$, $j, l \in \{1, 2, \ldots, m\}$, $\delta \in \{R, N, L\}$, q_i, q_k are states of TM, a_j, a_l - symbols of TM and δ - denotes the shift of the head of TM to the appropriate directions (R - to the right, N - no shift and L - to the left).

It is easy to proof the existence of self-describing Turing machines using recursive theory.

Let T_i be TM with Gödel number i and ϕ_i be the function, computed by T_i.

Let an argument 0 of the function ϕ_i corresponds to the blank tape of the machine T_i.

Theorem 1 (see [2]). *There exists a self-describing Turing machine, that is, there exists an integer n_0 such that T_{n_0} prints its own* *"description" n_0 on an initially blank tape:*

$$\phi_{n_0}(0) = n_0.$$

Proof. We use well known theorem from recursive theory about fixed point that states:

Let $\phi_0, \phi_1, \ldots,$ be any effective enumeration (Gödel numbering) of all computable functions, and let h be any total computable function. Then there exists a number i_0 such that

$$\phi_{i_0} = \phi_{h(i_0)}.$$

We call i_0 a fixed point *for h.*

Define h to be the function such that for each n, $h(n)$ equals an index of a machine that, when started with blank tape, prints out n and halts, i.e., such that $\phi_{h(n)}(0) = n$. Then let n_0 be a fixed point for h:

$$\phi_{n_0}(0) = \phi_{h(n_0)}(0) = n_0.$$

3 The structure of a self-describing Turing machine

Before going to the encoding, we now indicate the general structure of a self-describing Turing machine. It is a well-known structure, the reader can find it in the several papers, for instance [8, 9]. In the latter paper, the same plan is applied to cellular automata, which are self-reproducing. As we depart in some details from that plan, we indicate it, for the reader's conveniency.

Note. *A first way would be to impose a condition, observed by natural encodings: the encoding must be some morphism with respect to concatenation. Indeed, a Turing program is a finite sequence of instructions, and it is natural first to define the encoding of instructions and then to define the encoding of the program as a concatenation of the encodings of the instructions which constitutes the program.*

Let M be a self-describing Turing machine with code $< M >$. It is assumed that the starting configuration of M is the blank tape. The general idea consists in splitting the code in two parts:

$$< M >=< T >< W >$$

This splitting corresponds to a superposition of M itself in two machines, T – $''transcoder''$ and W – $''writer''$, in such a way that $< M > = < T >< W >$. In order to get a self-describing machine, it is enough to assume that W writes down $< T >$ and that T transforms $< T >$ into $< W >$.

It is plain that, in some way depending on the chosen encoding, $< W >$ is a simple function of $< T >$: this will be the case if, for this purpose, the encoding performs a concatenation-homomorphism on $< T >$.

But the implementation of this simple idea, makes the situation a bit more complex. The difficulty comes from the transition from the control of the computation by W to the control by T. As $< T >$ comes first, this means that a starting sub-machine, say S – $''starter''$, should be added in order to transfer the control of the computation to W. For an analogous reason, after $< W >$ comes a sub-machine transferring the control to T. As M should stop after T has performed its computation, another sub-machine, say $E - ''ender''$, should achieve the computation of M.

And so, we come to the splitting of M into five sub-machines:

$$M = S \circ T \circ E \circ W \circ J$$

When the *starter* and the *writer* have achieved the computation, the following configuration will appear on the tape:

$$< S >< T >< E > \square$$

The square at the end of the configuration focuses the attention on a delimiter, put down on the tape by the *writer* which signalizes to the *transcoder* the end of its data.

Then, the *transcoder* comes into action, and when its computation is achieved, the configuration is now:

$$< S >< T >< E > \Box < W >< J' >$$

The comparison with $< M >$, shows that the *ender* has to performs two tasks:
- erasing the delimiter,
- and completing the code of the *jumper*.

4 The principles of encoding

We introduce new principles of encoding. According them we design self-describing machines S_1, S_2 and S_3 with 275, 224 and 206 instructions accordingly. The main of these principles are: the ”position encoding”; using 4 symbols for encoding (Thatcher used 2 symbols); ”relative” addressing instead ”absolute” (therefore the sub-machine *jumper* becomes not necessary).

(I) We use the notion of *position encoding*, i.e. encoding depends on the place of encoded element among other encoded elements. We consider that all instructions of TM are in the linear ordering, such the instructions beginning with q_i precede the instructions beginning with q_{i+1} and instruction which beginning with a pair $< q_i, a_j >$, immediately precedes the instruction, which beginning with a pair $< q_i, a_{j+1} >$. This allows us to encode not the whole instruction $< q_i, a_j, a_l, \delta, q_k >$, but the correspondent triple $< a_l, \delta, q_k >$ only.

(II) We consider the Turing machines without stationary instructions, i.e. for all instructions $< q_i, a_j, a_l, \delta, q_k >$ must be $\delta \in \{R, L\}$. It is easy to proof, that for every Turing machine T there is a Turing machine T' without stationary instructions which models T.

(III) The triple $I_{i,j} =< a_l, \delta, q_k >$ $(k = (i \pm t) \textbf{ mod } n)$, corresponds to $< q_i, a_j, a_l, \delta, q_k >$, where $i, k \in \{1, \ldots, n\}$, $t \in \{1, \ldots, n-1\}$, $j, l \in \{1, \ldots, m\}$, $\delta \in \{R, L\}$, and it is encoded as follows: $c(I_{i,j}) = c(a_l)c(\delta)c(\pm t)2$, where $c(a_l)$ is a binary code (a word consists of 0's and 1's) of the symbol a_l, $c(\delta) = 0$, if $\delta = L$, and $c(\delta) = 1$, if $\delta = R$,

$c(\pm t) = c(\pm)c(t)$ is a binary code, corresponding to the state q_k and "2" is a special symbol (delimiter). Here $c(-) = 0$, $c(+) = 1$ and $c(t)$ is a binary expression of the number t. So we use the principle of a relative addressing instead of an absolute addressing, i.e. it encodes the difference between a real state of TM and a new state.

It is essential that the last instruction of the sub-machine *writer* has the type $< q_n, a_j, a_l, R, q_1 >$ (one also used as sub-machine *jumper*) and may be encoded like all other instructions of *writer* which have the same type $< q_i, a_j, a_l, R, q_{i+1} >$ (taking into account that $1 = (n+1) \textbf{ mod } n$). So this trick allows us really avoid the sub-machine *jumper* and considerably decrease the size of the self-describing Turing machine.

Let $R_i = I_{i,1}, I_{i,2}, \ldots, I_{i,m}$, where $I_{i,j}$ $(i = 1, \ldots, n,\ j = 1, \ldots, m)$ are triples, corresponding to the instructions, beginning with the state q_i. So the code for the group of triples R_i, corresponding the instructions, beginning with q_i, is as follows:

$$c(R_i) = c(I_{i,1})c(I_{i,2})\ldots c(I_{i,m})2$$

and the code of the whole program TM:

$$c(T) = c(R_1)c(R_2)\ldots c(R_n).$$

(IV) We try to minimize the size of the code $c(I_{i,j})$ and therefore consider the task of design of self-describing machine in the alphabet consisting of 4 symbols: 0, 1, 2, and b (blank symbol). We take into account informal analisys of design of a self-describing machines in the alphabet consisting of 2 and more symbols. Optimal variant is for the alphabet consisting of 4 symbols.

(V) The symbols of TM one encodes as follows:

c(b)=00, c(0)=01, c(1)=10, c(2)=11.

Let $|c(I_{i,j})|$ means the length of the code $c(I_{i,j})$, the analogous sense have the expressions $|c(q_k)|$, $|c(a_j)|$ and other.

We consider the special cases and rules of encoding:

	instruction	**code**
(1)	$q_i a_j a_j \delta q_i$	$c(I_{i,j}) = c(\delta)2;\ \lvert c(I_{i,j})\rvert = 2$
(2)	$q_i a_j a_j \delta q_{i+1}$	$c(I_{i,j}) = c(\delta)c(+)2;\ \lvert c(I_{i,j})\rvert = 3$
(3)	$q_i a_j a_l \delta q_k$ is absent	$c(I_{i,j}) = 002;\ \lvert c(I_{i,j})\rvert = 3$
(4)	$q_i a_j a_l \delta q_i$	$c(I_{i,j}) = c(a_l)c(\delta)2;\ \lvert c(I_{i,j})\rvert = 4$
(5)	$q_i a_j a_l \delta q_{i+1}$	$c(I_{i,j}) = c(a_l)c(\delta)c(+)2;\ \lvert c(I_{i,j})\rvert = 5$
(6)	$q_i a_j a_j \delta q_{i+t}$ $(t \leq 15)$	$c(I_{i,j}) = c(\delta)c(t)2;\ \lvert c(I_{i,j})\rvert = 6$
(7)	$q_i a_j a_l \delta q_{i+t}$ $(t \leq 7)$	$c(I_{i,j}) = c(a_l)c(\delta)c(t)2;\ \lvert c(I_{i,j})\rvert = 7$
(8)	$q_i a_j a_l \delta q_{i+t}$ $(t \leq 15)$	$c(I_{i,j}) = c(a_l)c(\delta)c(t)2;\ \lvert c(I_{i,j})\rvert = 8$
(9)	$q_i a_j a_l \delta q_{i\pm t}$ $(\lvert c(t)\rvert \geq 4)$	$c(I_{i,j}) = c(a_l)c(\delta)c(\pm)c(t)2;\ \lvert c(I_{i,j})\rvert \geq 9$
(10)	$R_i = I_{i,b}$	$c(R_i) = c(I_{i,b})2$
(11)	$R_i = I_{i,b}, I_{i,0}, I_{i,1}, I_{i,2}$	$c(R_i) = c(I_{i,b})c(I_{i,0})c(I_{i,1})c(I_{i,2})$

We note, that (9) is the general case, i.e. we may encode all instructions of TM according to this rule, but the size of instruction code will be longer, of course.

The encoding by above principles is called the encoding of type E_1.

5 The self-describing Turing machine S_1 with 275 instructions

The program of the machine S_1

The programs of the sub-machines *starter*, *transcoder* **and** *ender*.

q_1b2Lq_2	-	-	-
q_2bbRq_{17}	-	-	q_222Lq_3
q_3bbRq_4	q_300Lq_3	q_311Lq_3	q_322Lq_3
q_4b0Rq_5	q_40bLq_4	q_41bLq_6	q_422Lq_9
q_5bbRq_{14}	-	-	-
q_6b1Rq_7	-	-	-
q_7bbRq_8	-	-	-
q_8b1Rq_{12}	q_800Rq_8	q_811Rq_8	q_822Rq_8
q_9bbLq_9	q_900Rq_{10}	q_911Rq_{10}	q_922Lq_{10}
$q_{10}b2Rq_{12}$	$q_{10}00Rq_{11}$	$q_{10}11Rq_{11}$	$q_{10}21Rq_{19}$
$q_{11}b2Rq_{12}$	-	-	$q_{11}22Rq_{11}$

$q_{12}b0Rq_{16}$	-	-	$q_{12}2bRq_{13}$
$q_{13}b1Rq_{15}$	$q_{13}00Rq_{13}$	$q_{13}11Rq_{13}$	$q_{13}22Rq_{13}$
$q_{14}b0Rq_{15}$	$q_{14}00Rq_{14}$	$q_{14}11Rq_{14}$	$q_{14}22Rq_{14}$
$q_{15}b1Rq_{16}$	-	-	-
$q_{16}b1Rq_{17}$	-	-	-
$q_{17}b1Rq_{18}$	-	-	$q_{17}2bRq_{21}$
$q_{18}b2Rq_{19}$	-	-	-
$q_{19}b2Lq_{3}$	-	-	$q_{19}21Rq_{20}$
$q_{20}b2Rq_{20}$	-	-	-

The program of the sub-machine *writer* is presented in the **Appendix 1**.

We describe now the operation of the machine S_1.

The machine S_1 starts with the state q_1, records the symbol "2" on the blank tape and shifts on one cell to the left on the tape (instruction q_1b2Lq_2). Then, if it meets the blank cell, it comes back at the cell originally considered in the state q_{17}, where there is a symbol "2" (q_2bbRq_{17}). Then the symbol "2" is destroyed, the machine goes to the state q_{21} and the sub-machine *writer* begins its operation ($q_{17}2bRq_{21}$). Such an unusual behavior of the sub-machine *starter* (instead of transmitting the control immediately to the sub-machine *writer*, it records the symbol "2", then it destroys this symbol and only after this the control is transmitted to the sub-machine *writer*), will be explained below.

The sub-machine *writer* codes instructions from q_1 to q_{20}. After the work of the sub-machine *writer* the tape of the machine S_1 has the following form:

$\dots bb$ 110122 111112002002012 112020202 0111200020000102
001012 1100122 101122 1122 1011002121212 02112112012
111010211211210110012 1111200200212 0111002002002000112
1010102121212 01112121212 101122 101122 1011200200200011002
111122 11001000020020021O112 122 $\overset{\downarrow}{b}\, b \dots$

Here the pointer indicates the cell in consideration and the machine S_1 is in the state q_1. The last group of symbols "122", should be, certainly, is replaced with the code "11122" of the group of instructions R_{20}, as will be made later on. The blanks on a tape have not any significance, they are shown here only for clarity.

Further, to group of symbols from the right "22" with the help of the instruction $q_1 b2Lq_2$ records more one symbol "2" (it is obtained "222") and the head is shifted to the left. (The group "222" serves as a "delimiter" for the sub-machine *transcoder*, it limits a file of symbols, which is interpreted by the sub-machine *transcoder* and serves a signal for a closing-up of the machine S_1). After this machine S_1 goes to the state q_3 and the sub-machine *transcoder* ($q_2 22Lq_3$) begins its work.

Now we can explain non-standard behavior of the sub-machine *starter*. All this is connected with liquidation of the sub-machine *jumper* and as a consequence, the sub-machine *ender* should not record at the end of the tape the code of the sub-machine *jumper*. As it was already spoken, the role of the sub-machine *jumper* is to execute the last instruction of the sub-machine *writer*, just the instruction $q_{251} b2Rq_1$, which *transcoder* interprets in a standard way. The difficulty is that now, after execution of the instruction $q_{251} b2Rq_1$, the sub-machine *transcoder* begins working, (*writer* does not begin working), as it follows from a sense of the state q_1. Therefore *starter* checks up, which symbol is at the left on the tape? If this symbol is the blank symbol "b", it means that it is necessary to go to the sub-machine *writer*, if there is the symbol "2", it means to go to the sub-machine *transcoder*, and the symbol "2" to the right end of a tape is thus attributed, by this "delimiter 222" is obtained.

The sense of the operation of the sub-machine *transcoder* consists in the following. The machine writes a *mark* on the tape (it is the symbol "b") and shifts it from the left to the right. Thus it stores a symbol, replaced by the *mark*, and writes a code of the instruction of the sub-machine *writer*, corresponding to this symbol, at the right end on the tape. If this stored symbol is "0", it records the code "011122" on the tape (rules (5), (11)), if this symbol is "1", it records the code "101122" (those rules (5), (11)), if this symbol is "2", it records the code "111122" (rule (5), (11) again).

We consider the operation of *transcoder* on an example. Let the tape of the machine S_1 in an operating time *transcoder* has a form:

$$\ldots a_{L,1} \overset{\downarrow}{b} 1 a_{R_1,1} \ldots a_{R_1,t_1} 1222 a_{R_2,1} \ldots a_{R_2,t_2} bb \ldots$$

and the machine is in the state q_3. With the help of instructions q_3bbRq_4, q_41bLq_6, q_6b1Rq_7, q_7bbRq_8 the *mark* b is shifted one cell to the right, thus the machine stores a symbol "1". Further, the machine is shifted to the right at the end of the tape (instructions q_800Rq_8, q_811Rq_8, q_822Rq_8) and records the code of the instruction of the submachine *writer*, which corresponds to the stored symbol "1" on the tape. It is made with the help of instructions q_8b1Rq_{12}, $q_{12}b0Rq_{16}$, $q_{16}b1Rq_{17}$, $q_{17}b1Rq_{18}$, $q_{18}b2Rq_{19}$ and $q_{19}b2Lq_3$. The tape of the machine at this moment will have the form:

$$\ldots a_{L,1} 1 b a_{R_1,1} \ldots a_{R_1,t_1} 1222 a_{R_2,1} \ldots a_{R_2,t_2} 1011 \overset{\downarrow}{2} 2bb \ldots$$

and the machine is in the state q_3. Further the machine goes into process of a search the *mark* b on the tape at the left and the process is repeated.

We consider in detail, as process of a closing-up of the machine S_1 occurs. The tape of the machine at this moment has the form:

$$\ldots a_{L,1} 122 \overset{\downarrow}{b} 2 a_{R,1} \ldots a_{R,t_1} bb \ldots$$

and the machine is in the state q_3. For simplicity we record this configuration as follows: $\ldots 122q_3b2 \ldots$

We consider the operation of the machine S_1 step by step:

$$\begin{array}{l}
\ldots 122q_3b2 \ldots \\
\ldots 122bq_42 \ldots \\
\ldots 122q_9b2 \ldots \\
\ldots 12q_92b2 \ldots \\
\ldots 1q_{10}22b2 \ldots \\
\ldots 11q_{19}2b2 \ldots \\
\ldots 111q_{20}b2 \ldots \\
\ldots 1112q_{20}2 \ldots
\end{array}$$

and the machine S_1 stops, as far as there is no instruction for the pair $< q_{20}, 2 >$, thus having restored code "11122" of group of instructions R_{20}.

Now on the tape of the machine S_1 there is the description of its internal structure and the size of this machine is 275 instructions (46 instructions of the sub-machines *starter*, *transcoder*, *ender* and plus 229 instructions of the sub-machine *writer*). So, the theorem is proven:

Theorem 2. *There exists the self-describing Turing machine with 275 instructions, which satisfies to the coding of type* E_1 .

6 The self-describing Turing machine S_2 with 224 instructions

The further reduction in the number of instructions of a self-describing Turing machine is possible thanks to the fact of some agreements and changes in the sub-machines *starter*, *transcoder*, *ender* and, hence, in the sub-machine *writer* of the machine S_1.

(i) We consider, that TM can start with any state (not only with q_1). This technique will allow us to reduce the size of self-describing machine.

(ii) The coding of programs of TM admits "noise", i.e. in a body of the code, sequences of symbols, which do not relate to the code, can be contained, which, however, are unequivocally distinguished and do not infringe principles of unequivocal encoding and decoding.

We make also an addition to the specific cases of coding (V):

	instruction	**code**
(2')	$q_i a_j a_j R q_{i-1}$	$c(I_{i,j}) = c(R)c(-)2 = 102;\ \lvert c(I_{i,j})\rvert = 3$
(11')	$R_i = I_{i,b}, I_{i,0}, I_{i,1}$	$c(R_i) = c(I_{i,b})c(I_{i,0})c(I_{i,1})2$

The encoding of type E_1, which follows additional conditions (i), (ii), (2") and (11"), is the encoding of type E_2.

The program of the machine S_2

The programs of the sub-machines *starter*, *transcoder* and *ender*.

q_1b2Lq_2	-	-	-
q_2bbRq_3	q_200Lq_2	q_211Lq_2	q_222Lq_2
q_3b0Rq_4	q_30bLq_3	q_31bLq_5	q_322Lq_8
q_4bbRq_{13}	-	-	-
q_5b1Rq_6	-	-	-
q_6bbRq_7	-	-	-
q_7b1Rq_{11}	q_700Rq_7	q_711Rq_7	q_722Rq_7
q_8bbLq_8	q_800Rq_9	q_811Rq_9	q_822Lq_9
q_9b2Rq_{11}	q_900Rq_{10}	q_911Rq_{10}	-
$q_{10}bbRq_{19}$	-	-	$q_{10}22Rq_9$
$q_{11}b0Rq_{15}$	-	-	$q_{11}2bRq_{12}$
$q_{12}b1Rq_{14}$	$q_{12}00Rq_{12}$	$q_{12}11Rq_{12}$	$q_{12}22Rq_{12}$
$q_{13}b0Rq_{14}$	$q_{13}00Rq_{13}$	$q_{13}11Rq_{13}$	$q_{13}22Rq_{13}$
$q_{14}b1Rq_{15}$	-	-	-
$q_{15}b1Rq_{16}$	-	-	-
$q_{16}b1Rq_{17}$	-	-	-
$q_{17}b2Rq_{18}$	-	-	-
$q_{18}b2Lq_2$	-	-	-

The program of the sub-machine *writer* is presented in the **Appendix 2**.

We describe now the operation of the machine S_2.

The machine S_2, as distinct from the machine S_1, starts with the state q_{10}, immediately passing the control to the sub-machine *writer*.

After the work the sub-machine *writer* the tape of the machine S_2 has the following form:

$\ldots bb$ 110122 112020202 0111200020000102001012 1100122
101122 1122 1011002121212 02112112012 11101021121122
110012002002102 011100200200200112 1010102121212
01112121212 101122 101122 101122 111122 11001000022 $\overset{\downarrow}{b}\, b\ldots$

Here the pointer indicates the cell in consideration and the machine is in the state q_1 after execution of the instruction $q_{204}b2Rq_1$ of the sub-machine *writer*.

Further, to group of symbols from the right "22" with the help of the instruction q_1b2Lq_2 appends more one symbol "2" (thus "222" is obtained) and the head is shifted to the left. As well as for the machine S_1, the group "222" serves a "delimiter" for the sub-machine *transcoder*, it limits a file of symbols, which is interpreted by the sub-machine *transcoder* and serves as a signal for a closing-up of the machine S_2. After this machine S_2 goes to the state q_2 and the sub-machine *transcoder* (q_222Lq_2) begins its work.

We consider in detail, as process of a closing-up of the machine S_2 occurs. The tape of the machine at this moment has the form:

$$\dots a_{L,1}22\overset{\downarrow}{b}2a_{R,1}\dots a_{R,t_1}bb\dots$$

and the machine is in the state q_2. For simplicity we record this configuration as follows: $\dots 22q_2b2\dots$

We consider the operation of the machine S_1 step by step:

$$\begin{array}{l} \dots 22q_2b2\dots \\ \dots 22bq_32\dots \\ \dots 22q_8b2\dots \\ \dots 2q_82b2\dots \\ \dots q_922b2\dots \end{array}$$

and the machine S_2 stops, as far as there is no instruction for the pair $< q_9, 2 >$. Thus on a tape there is a "noise"("dust"): two symbols "b2" after the code of group of instructions R_{18} and before the code of the first instruction of the sub-machine *writer*. Obviously, that presence of this "dust" will not affect the results of decoding.

Now on the tape of the machine S_2 there is the description of its internal structure and a size of this machine is 224 instructions (40 instructions of the sub-machine *starter*, *transcoder*, *ender* and plus 184 instructions of the sub-machine *writer*). So, the theorem is proven:

Theorem 3. *There exists self-describing Turing machines with 224 instructions, which satisfies to the coding of type E_2.*

7 The self-describing Turing machine S_3 with 206 instructions

The further reduction in the number of instructions of a self-describing Turing machine is possible thanks to the fact of some agreements about coding and of respective alterations in the sub-machine *writer* of the machine S_2.

We make a new addition to specific cases of coding (V):

	instruction	**code of S_2**	**code of S_3**
(12)	$q_i a_j 2 L q_{i-16}$	1100100002	00002
(13)	$q_i a_j b L q_{i+2}$	0000102	00102
(14)	$q_i a_j a_j L q_{i+5}$	001012	01002
(15)	$q_i a_j a_j R q_{i+9}$	110012	01102
(16)	$q_i a_j 1 R q_{i+4}$	1011002	10002
(17)	$q_i a_j 2 R q_{i+2}$	1110102	10102
(18)	$q_i a_j 0 R q_{i+4}$	0111002	11002
(19)	$q_i a_j 1 R q_{i+2}$	1010102	11102

The encoding of type E_2, which follows additions (12) - (19), is an encoding of type E_3.

The machine S_3

The sub-machines *starter*, *transcoder* and *ender* are the same as in the machine S_2.

The sub-machine *writer* of the machine S_2 is changed conform the rules (12) - (19). The rule (12) reduces a number of instructions *writer* on 5 (instruction $q_{18}b2Lq_2$), (13) on 2 ($q_3 1bLq_5$), (14) on 1 ($q_3 22Lq_8$), (15) on 2 ($q_4 bbRq_{13}$ and $q_{10} bbRq_{19}$), (16) on 2 ($q_7 b1Rq_{11}$), (17) on 2 ($q_9 b2Rq_{11}$), (18) on 2 ($q_{11} b0Rq_{15}$) and (19) on 2 ($q_{12} b1Rq_{14}$).

The total number of instructions, by which the size of the sub-machine *writer* decreases, equals 18. So, the theorem is proven:

Theorem 4. *There exists a self-describing Turing machine with 206 instructions, which satisfies to the coding of type E_3.*

Acknowledgments

Several fundings created the conditions of a fruitfull collaborations between the two authors, which made them possible to do this joint work. In this respect, the first and the second authors thank both the French Ministery for Education, Universities and Research and *Institut Universitaire de Technologie* of Metz (France).

References

[1] M.Arbib, Simple Self-Reproducing Universal Automata. - Information and Control, v.9, 1966, pp.177 - 189.

[2] M.Arbib, Brains, machines and mathematics. - Springer-Verlag, 1987.

[3] John Case, A note on Degrees of Self-Describing Turing Machines. - JACM, v. 18, No.3, 1971, pp.329-338.

[4] C.Y.Lee, A Turing machine which prints its own code script. - In Proc. Symp. Math. Theory of Automata, Polytechnic Press, Brooklyn, New York, 1963, pp.155-164.

[5] M.Margenstern, Yu.Rogozhin, About Turing machines reproducing their own code. - CAW'97, Cellular Automata Workshop, Abstracts, Gargnano (Italy), September, 1997.

[6] J.von Newmann, The general and logical theory of automata. In Cerebral Mechanisms in Behavior - The Nixon Symposium, pp.1-41, Pasadena, 1951 (Also in: John von Newmann, Collected Works, Pergamon, Oxford, 1963, 288-328).

[7] J. von Newmann, Theory of Self-Reproducing Automata. - edited and completed by A.W.Burks, Urbana: University of Illinois Press, 1966.

[8] J.W.Thatcher. The Construction of a Self-Describing Turing Machines. - In Proc. Symp. Math. Theory of Automata, Polytechnic Press, Brooklyn, New York, 1963, pp.165-171.

[9] J.W.Thatcher, Self-describing Turing machines and self-reproducing cellular automata. - In Essays on Cellular Automata (A.W.Burks, Ed.) University of Illinois Press, 1970, pp.103-131.

[10] A.M.Turing, On computable Numbers, with an Application to the Entscheidungs Problem. - Proc. London Math. Soc., v.24, 1936.

[11] H.Wang, A Variant to Turing's theory of Computing machines. - JACM,vol.4, 1957.

Appendix 1

The sub-machine *writer*

The submachine *writer* is very simple, it records in succession the codes of the instructions from q_1 to q_{20} and consists only of instructions of the kind $< q_i, b, a_j, R, q_{i+1} >$. The instructions for pairs $< q_i, 0 >$, $< q_i, 1 >$ and $< q_i, 2 >$ are absent. In the first column are recorded the instructions of the sub-machines *starter*, *transcoder* and *ender* which are coded with the help of *writer*, in the second - the code, in the third - the number of appropriate rules of formation of the code, and in the fourth - appropriate instructions of the sub-machine *writer*.

q_1b2Lq_2	110122	(5), (10)	$q_{21}b1Rq_{22}$
-			$q_{22}b1Rq_{23}$
-			$q_{23}b0Rq_{24}$
-			$q_{24}b1Rq_{25}$
			$q_{25}b2Rq_{26}$
			$q_{26}b2Rq_{27}$

q_2bbRq_{17}	111112	(6)	$q_{27}b1Rq_{28}$
-	002	(3)	$q_{28}b1Rq_{29}$
-	002	(3)	$q_{29}b1Rq_{30}$
q_222Lq_3	012	(2), (11)	$q_{30}b1Rq_{31}$
			$q_{31}b1Rq_{32}$
			$q_{32}b2Rq_{33}$
			$q_{33}b0Rq_{34}$
			$q_{34}b0Rq_{35}$
			$q_{35}b2Rq_{36}$
			$q_{36}b0Rq_{37}$
			$q_{37}b0Rq_{38}$
			$q_{38}b2Rq_{39}$
			$q_{39}b0Rq_{40}$
			$q_{40}b1Rq_{41}$
			$q_{41}b2Rq_{42}$
q_3bbRq_4	112	(2)	$q_{42}b1Rq_{43}$
q_300Lq_3	02	(1)	$q_{43}b1Rq_{44}$
q_311Lq_3	02	(1)	$q_{44}b2Rq_{45}$
q_322Lq_3	02	(1), (11)	$q_{45}b0Rq_{46}$
			$q_{46}b2Rq_{47}$
			$q_{47}b0Rq_{48}$
			$q_{48}b2Rq_{49}$
			$q_{49}b0Rq_{50}$
			$q_{50}b2Rq_{51}$
q_4b0Rq_5	01112	(5)	$q_{51}b0Rq_{52}$
q_40bLq_4	0002	(4)	$q_{52}b1Rq_{53}$
q_41bLq_6	0000102	(7)	$q_{53}b1Rq_{54}$
q_422Lq_9	001012	(6), (11)	$q_{54}b1Rq_{55}$
			$q_{55}b2Rq_{56}$
			$q_{56}b0Rq_{57}$
			$q_{57}b0Rq_{58}$
			$q_{58}b0Rq_{59}$
			$q_{59}b2Rq_{60}$
			$q_{60}b0Rq_{61}$
			$q_{61}b0Rq_{62}$

			$q_{62}b0Rq_{63}$
			$q_{63}b0Rq_{64}$
			$q_{64}b1Rq_{65}$
			$q_{65}b0Rq_{66}$
			$q_{66}b2Rq_{67}$
			$q_{67}b0Rq_{68}$
			$q_{68}b0Rq_{69}$
			$q_{69}b1Rq_{70}$
			$q_{70}b0Rq_{71}$
			$q_{71}b1Rq_{72}$
			$q_{72}b2Rq_{73}$
q_5bbRq_{14}	1100122	(6), (10)	$q_{73}b1Rq_{74}$
-			$q_{74}b1Rq_{75}$
-			$q_{75}b0Rq_{76}$
-			$q_{76}b0Rq_{77}$
			$q_{77}b1Rq_{78}$
			$q_{78}b2Rq_{79}$
			$q_{79}b2Rq_{80}$
q_6b1Rq_7	101122	(5), (10)	$q_{80}b1Rq_{81}$
-			$q_{81}b0Rq_{82}$
-			$q_{82}b1Rq_{83}$
-			$q_{83}b1Rq_{84}$
			$q_{84}b2Rq_{85}$
			$q_{85}b2Rq_{86}$
q_7bbRq_8	1122	(2), (10)	$q_{86}b1Rq_{87}$
-			$q_{87}b1Rq_{88}$
-			$q_{88}b2Rq_{89}$
-			$q_{89}b2Rq_{90}$
q_8b1Rq_{12}	1011002	(7)	$q_{90}b1Rq_{91}$
q_800Rq_8	12	(1)	$q_{91}b0Rq_{92}$
q_811Rq_8	12	(1)	$q_{92}b1Rq_{93}$
q_822Rq_8	12	(1), (11)	$q_{93}b1Rq_{94}$
			$q_{94}b0Rq_{95}$
			$q_{95}b0Rq_{96}$
			$q_{96}b2Rq_{97}$

			$q_{97}b1Rq_{98}$
			$q_{98}b2Rq_{99}$
			$q_{99}b1Rq_{100}$
			$q_{100}b2Rq_{101}$
			$q_{101}b1Rq_{102}$
			$q_{102}b2Rq_{103}$
q_9bbLq_9	02	(1)	$q_{103}b0Rq_{104}$
q_900Rq_{10}	112	(2)	$q_{104}b2Rq_{105}$
q_911Rq_{10}	112	(2)	$q_{105}b1Rq_{106}$
q_922Lq_{10}	012	(2), (11)	$q_{106}b1Rq_{107}$
			$q_{107}b2Rq_{108}$
			$q_{108}b1Rq_{109}$
			$q_{109}b1Rq_{110}$
			$q_{110}b2Rq_{111}$
			$q_{111}b0Rq_{112}$
			$q_{112}b1Rq_{113}$
			$q_{113}b2Rq_{114}$
$q_{10}b2Rq_{12}$	1110102	(7)	$q_{114}b1Rq_{115}$
$q_{10}00Rq_{11}$	112	(2)	$q_{115}b1Rq_{116}$
$q_{10}11Rq_{11}$	112	(2)	$q_{116}b1Rq_{117}$
$q_{10}21Rq_{19}$	10110012	(8), (11)	$q_{117}b0Rq_{118}$
			$q_{118}b1Rq_{119}$
			$q_{119}b0Rq_{120}$
			$q_{121}b2Rq_{122}$
			$q_{122}b1Rq_{123}$
			$q_{123}b1Rq_{124}$
			$q_{124}b2Rq_{125}$
			$q_{125}b1Rq_{126}$
			$q_{126}b1Rq_{127}$
			$q_{127}b2Rq_{128}$
			$q_{128}b1Rq_{129}$
			$q_{129}b0Rq_{130}$
			$q_{130}b1Rq_{131}$
			$q_{131}b1Rq_{132}$
			$q_{132}b0Rq_{133}$

			$q_{133}b0Rq_{134}$
			$q_{134}b1Rq_{135}$
			$q_{135}b2Rq_{136}$
$q_{11}b2Rq_{12}$	11112	(5)	$q_{136}b1Rq_{137}$
-	002	(3)	$q_{137}b1Rq_{138}$
-	002	(3)	$q_{138}b1Rq_{139}$
$q_{11}22Rq_{11}$	12	(1), (11)	$q_{139}b1Rq_{140}$
			$q_{140}b2Rq_{141}$
			$q_{141}b0Rq_{142}$
			$q_{142}b0Rq_{143}$
			$q_{143}b2Rq_{144}$
			$q_{144}b0Rq_{145}$
			$q_{145}b0Rq_{146}$
			$q_{146}b2Rq_{147}$
			$q_{147}b1Rq_{148}$
			$q_{148}b2Rq_{149}$
$q_{12}b0Rq_{16}$	0111002	(7)	$q_{150}b0Rq_{151}$
-	002	(3)	$q_{151}b1Rq_{152}$
-	002	(3)	$q_{152}b1Rq_{153}$
$q_{12}2bRq_{13}$	00112	(5), (11)	$q_{153}b1Rq_{154}$
			$q_{154}b0Rq_{155}$
			$q_{155}b0Rq_{156}$
			$q_{156}b2Rq_{157}$
			$q_{157}b0Rq_{158}$
			$q_{158}b0Rq_{159}$
			$q_{159}b2Rq_{160}$
			$q_{160}b0Rq_{161}$
			$q_{161}b0Rq_{162}$
			$q_{162}b2Rq_{163}$
			$q_{163}b0Rq_{164}$
			$q_{164}b0Rq_{165}$
			$q_{165}b1Rq_{166}$
			$q_{166}b1Rq_{167}$
			$q_{167}b2Rq_{168}$
$q_{13}b1Rq_{15}$	1010102	(7)	$q_{168}b1Rq_{169}$

$q_{13}00Rq_{13}$	12	(1)	$q_{169}b0Rq_{170}$
$q_{13}11Rq_{13}$	12	(1)	$q_{170}b1Rq_{171}$
$q_{13}22Rq_{13}$	12	(1), (11)	$q_{171}b0Rq_{172}$
			$q_{172}b1Rq_{173}$
			$q_{173}b0Rq_{174}$
			$q_{174}b2Rq_{175}$
			$q_{175}b1Rq_{176}$
			$q_{176}b2Rq_{177}$
			$q_{177}b1Rq_{178}$
			$q_{178}b2Rq_{179}$
			$q_{179}b1Rq_{180}$
			$q_{180}b2Rq_{181}$
$q_{14}b0Rq_{15}$	01112	(5)	$q_{181}b0Rq_{182}$
$q_{14}00Rq_{14}$	12	(1)	$q_{182}b1Rq_{183}$
$q_{14}11Rq_{14}$	12	(1)	$q_{183}b1Rq_{184}$
$q_{14}22Rq_{14}$	12	(1), (11)	$q_{184}b1Rq_{185}$
			$q_{185}b2Rq_{186}$
			$q_{186}b1Rq_{187}$
			$q_{187}b2Rq_{188}$
			$q_{188}b1Rq_{189}$
			$q_{189}b2Rq_{190}$
			$q_{190}b1Rq_{191}$
			$q_{191}b2Rq_{192}$
$q_{15}b1Rq_{16}$	101122	(4), (10)	$q_{192}b1Rq_{193}$
-			$q_{193}b0Rq_{194}$
-			$q_{194}b1Rq_{195}$
-			$q_{195}b1Rq_{196}$
			$q_{196}b2Rq_{197}$
			$q_{197}b2Rq_{198}$
$q_{16}b1Rq_{17}$	101122	(4), (10)	$q_{198}b1Rq_{199}$
-			$q_{199}b0Rq_{200}$
-			$q_{200}b1Rq_{201}$
-			$q_{201}b1Rq_{202}$
			$q_{202}b2Rq_{203}$
			$q_{203}b2Rq_{204}$

$q_{17}b1Rq_{18}$	10112	(5)	$q_{204}b1Rq_{205}$
-	002	(3)	$q_{205}b0Rq_{206}$
-	002	(3)	$q_{206}b1Rq_{207}$
$q_{17}2bRq_{21}$	0011002	(7), (11)	$q_{207}b1Rq_{208}$
			$q_{208}b2Rq_{209}$
			$q_{209}b0Rq_{210}$
			$q_{210}b0Rq_{211}$
			$q_{211}b2Rq_{212}$
			$q_{212}b0Rq_{213}$
			$q_{213}b0Rq_{214}$
			$q_{214}b2Rq_{215}$
			$q_{215}b0Rq_{216}$
			$q_{216}b0Rq_{217}$
			$q_{217}b1Rq_{218}$
			$q_{218}b1Rq_{219}$
			$q_{219}b0Rq_{220}$
			$q_{220}b0Rq_{221}$
			$q_{221}b2Rq_{222}$
$q_{18}b2Rq_{19}$	111122	(5), (10)	$q_{222}b1Rq_{223}$
-			$q_{223}b1Rq_{224}$
-			$q_{224}b1Rq_{225}$
-			$q_{225}b1Rq_{226}$
			$q_{226}b2Rq_{227}$
			$q_{227}b2Rq_{228}$
$q_{19}b2Lq_{3}$	1100100002	(9)	$q_{228}b1Rq_{229}$
-	002	(3)	$q_{229}b1Rq_{230}$
-	002	(3)	$q_{230}b0Rq_{231}$
$q_{19}21Rq_{20}$	10112	(5), (11)	$q_{231}b0Rq_{232}$
			$q_{232}b1Rq_{233}$
			$q_{233}b0Rq_{234}$
			$q_{234}b0Rq_{235}$
			$q_{235}b0Rq_{236}$
			$q_{236}b0Rq_{237}$
			$q_{237}b2Rq_{238}$
			$q_{238}b0Rq_{239}$

			$q_{239}b0Rq_{240}$
			$q_{240}b2Rq_{241}$
			$q_{241}b0Rq_{242}$
			$q_{242}b0Rq_{243}$
			$q_{243}b2Rq_{244}$
			$q_{244}b1Rq_{245}$
			$q_{245}b0Rq_{246}$
			$q_{246}b1Rq_{247}$
			$q_{247}b1Rq_{248}$
			$q_{248}b2Rq_{249}$
$q_{20}b2Rq_{20}$	122	(4), (10)	$q_{249}b1Rq_{250}$
-			$q_{250}b2Rq_{251}$
-			$q_{251}b2Rq_1$

Appendix 2

The sub-machine *writer*

As well as in the case of the machine S_1, in the first column one records instructions of the sub-machines *starter*, *transcoder* and *ender*, which are coded with the help *writer*, in the second - the code and in the third - the number of appropriate rules of formation of the code. We shall not record obvious instructions of the sub-machine *writer* in view of simplicity of their generation and for brevity.

q_1b2Lq_2	110122	(5), (10)
-		
-		
-		
q_2bbRq_3	112	(2)
q_200Lq_2	02	(1)
q_211Lq_2	02	(1)
q_222Lq_2	02	(1), (11)

q_3b0Rq_4	01112	(5)
q_30bLq_3	0002	(4)
q_31bLq_5	0000102	(7)
q_322Lq_8	001012	(6), (11)
q_4bbRq_{13}	1100122	(6), (10)
-		
-		
-		
q_5b1Rq_6	101122	(5), (10)
-		
-		
-		
q_6bbRq_7	1122	(2), (10)
-		
-		
-		
q_7b1Rq_{11}	1011002	(7)
q_700Rq_7	12	(1)
q_711Rq_7	12	(1)
q_722Rq_7	12	(1), (11)
q_8bbLq_8	02	(1)
q_800Rq_9	112	(2)
q_811Rq_9	112	(2)
q_822Lq_9	012	(2), (11)
q_9b2Rq_{11}	1110102	(7)
q_900Rq_{10}	112	(2)
q_911Rq_{10}	112	(2)
-	2	(11')

$q_{10}bbRq_{19}$	110012	(6)
-	002	(3)
-	002	(3)
$q_{10}22Rq_{9}$	102	(2'), (11)
$q_{11}b0Rq_{15}$	0111002	(7)
-	002	(3)
-	002	(3)
$q_{11}2bRq_{12}$	00112	(5), (11)
$q_{12}b1Rq_{14}$	1010102	(7)
$q_{12}00Rq_{12}$	12	(1)
$q_{12}11Rq_{12}$	12	(1)
$q_{12}22Rq_{12}$	12	(1), (11)
$q_{13}b0Rq_{14}$	01112	(5)
$q_{13}00Rq_{13}$	12	(1)
$q_{13}11Rq_{13}$	12	(1)
$q_{13}22Rq_{13}$	12	(1), (11)
$q_{14}b1Rq_{15}$	101122	(5), (10)
-		
-		
-		
$q_{15}b1Rq_{16}$	101122	(5), (10)
-		
-		
-		
$q_{16}b1Rq_{17}$	101122	(5), (10)
-		
-		
-		

$q_{17}b2Rq_{18}$	111122	(5), (10)
-		
-		
-		
$q_{18}b2Lq_{2}$	11001000022	(9), (10)
-		
-		
-		

Information about authors:

Maurice Margenstern, Yurii Rogozhin,
Received December 19, 1997

Prof. Maurice Margenstern
Université de Metz, I.U.T. de Metz,
Département d'Informatique,
Île du Saulcy, 57045 Metz Cedex, FRANCE
e-mail: *margens@iut.univ-metz.fr*

Dr. Yurii Rogozhin
Institute of mathematics,
Academy of Sciences of Moldova,
5 Academiei str., Kishinev,
MD-2028, Moldova
email: *rogozhin@cc.acad.md*