Decidability and Undecidability of the Halting Problem on Turing Machines, a Survey

Maurice Margenstern

I.U.T. de Metz, Département d'Informatique,
Île du Saulcy,
57045 Metz Cedex, France,
L.I.A.F.A. (Université Paris 7),
L.R.I.M. (Université de Metz),
e-mail: margens@iut.univ-metz.fr

Abstract. The paper surveys the main results obtained for Turing machines about the frontier between a decidable halting problem and universality. The notion of *decidability criterion* is introduced. Techniques for decidability proofs and for contructing universal objects are sketchily explained. A new approach for finding very small universal Turing machines is considered in the last part of the paper.

1 Introduction

Let $\mathcal{M}$ be the set of deterministic Turing machines with one head and one tape, infinite on both sides. Considerations of our survey only take place in $\mathcal{M}$. One knows that the halting problem for Turing machines in $\mathcal{M}$ is undecidable. This was settled by Turing, see [20], in the paper of 1936 in which he defined the machines later called after his name and in which he also proved the existence of universal Turing machines.

It should be noticed that Turing's undecidability theorem is proved by a simple diagonal argument, using the assumption that the set of *all* Turing machines in $\mathcal{M}$ is taken under consideration. What happens if only a *subset* of machines is considered?

The same problem is still undecidable when it is restricted to machines with alphabet $\{0, 1\}$ as the machine alphabet. In this case, the argument involves a universal Turing machine in the considered set of machines, say $\mathcal{S}$, which allows to reduce the general halting problem to the halting problem on $\mathcal{S}$.

1.1 A bit of history

Since the sixties and the early seventies, some research has been developed in the following direction, leading to partial answers which are far from closing the subject.

Identify any point $s \times l$ – take natural numbers s and l as coordinates – with the set of all Turing machines in $\mathcal{M}$ whose program contains s states, except the halting state(s), and uses l letters, including the blank symbol. Let us call such

a point *decidable* if the halting problem is decidable for any Turing machine in the corresponding set, and *undecidable* if the halting problem restricted to this set of machines is undecidable. As a matter of fact, for large enough s and l, it is easy to construct a universal machine occurring in the set attached to the point $s \times l$. By restricting the number of states or the alphabet of the machine, it is easily seen that any universal machine belonging to $x \times y$ set implies that for any n and m, with $n \geq x$ and $m \geq y$, the point $n \times m$ is undecidable too. Then it comes as a natural thing to wonder what are the smallest values of s and l for which the point $s \times l$ is undecidable.

An account of the results obtained up to the last seventies can be found in [15]. However, the best results were established in 1982, see [16], by Yuri Rogozhin who was the only one to improve them later on, see [17] and [18]. Yet his paper of 1982 had remained unnoticed by the scientific community during ten years. The technique used in order to obtain these results will be explained in our survey, in subsection 4.1.

Let's have a look now in the opposite direction: it is trivial that Turing machines on a single letter alphabet have a decidable halting problem. It is not that trivial that the same property holds for machines with a single state, whatever the number of symbols is in the machine alphabet. This was definitely proved by Hermann in 1966, see [2].

For more than one state, one does not know much. As for proving the decidability of the halting problem, M. Minsky writes in his famous book of 1967 that he and one of his students "did this for all 2×2 machines [1961, unpublished] by a tedious reduction to thirty-odd cases (unpublishable)", see [10], p. 281, last two lines. Six years later, Liudmila Pavlotskaya proved the same theorem, see [11], in a compact, very short proof. This paper has also remained unnoticed for many years. A few years later, see [13], she proved that the point 3×2 is also decidable.

All these results about the decidability and undecidability of the halting problem for Turing machines can be represented on figure 1, below.

1.2 Decidability criteria

Let us now define the notion of *decidability criterion*. Let c be an integer valued function defined on a set M of Turing machines in $\mathcal{M}$ with the following property: there is an integer f such that the halting problem is decidable for any machine $T \in M$ such that $c(T) < f$, and for any $k \geq f$ a universal machine $U \in M$ such that $c(U) = k$ can always be constructed. In that case, c is called *decidability criterion* and f is called its *frontier value*, see [5].

Shannon's result, see [19], shows that starting from two letters for the machine alphabet, it is always possible to construct a universal Turing machine. Thus the number of symbols of the machine alphabet, whatever the number of states is, provides a simple example of a decidability criterion. The same paper by Shannon and paper [2] show that the number of states, whatever the number of symbols is, also provides a decidability criterion with again two as a frontier value.

If we look closer at figure 1, below, we see that for Turing machines with only two states, the number of symbols provides a function that has not yet been proved to be a decidability criterion. Of course, the frontier value does exist. But the actual value is still unknown. Figure 1 shows that it lies somewhere between 4 and 18.

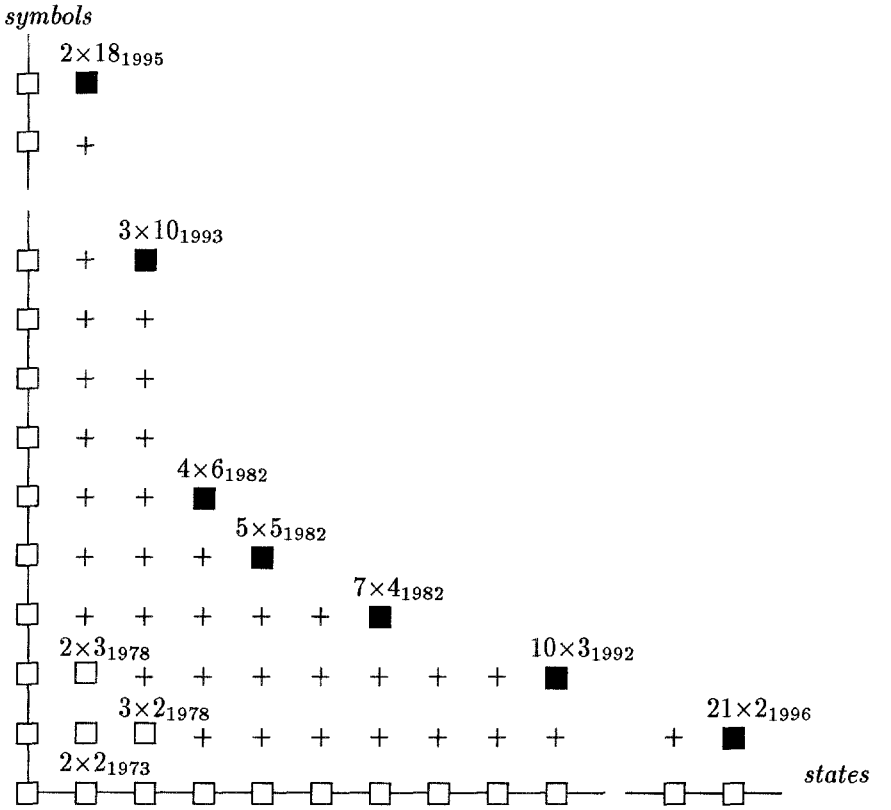


Fig. 1. The present state of the art

Black squares indicate undecidable points, white squares indicate decidable points, points marked with plus between the white squares lines and the line of black squares in the shape of a hyperbola correspond to sets for which the status of the halting problem is not known.

In our survey, we shall consider two other criteria, the only ones that have so far been established for Turing machines on alphabet $\{0, 1\}$.

One of them is the number of *colours* in the program of a Turing machine, see [4], where this number is defined as follows. The *colour* of an instruction of the program of Turing machine T is defined as the *projection* of the 5-tuple which

encodes the instruction – input state, input symbol, move, output symbol, new state – on the triple which consists in the input symbol, the move and the output symbol. In other words, the colour is precisely what can be seen of the tape by an observer to whom the internal states of the machine are hidden when instructions are performed. The number of colours is the cardinality of the projection of the whole machine program.

The other criterion is the laterality number of a Turing machine. In this case, we consider the projection which associates its move to each instruction. Then it can easily be seen that stationnary instructions can be associated to an ultimate true move on the left or on the right, unless a trivial cycling on a single cell occurs. This allows to extend the projection to all instructions with range in $L, R - L$ for move to the left, R for move to the right. Call this extension *laterality* of the instruction. The whole program is thus divided in two sets: instructions with L as laterality, instructions with R as laterality. The *laterality number* is the smallest cardinal of these two sets.

In the first section, we introduce the results. Then, in a second part, we shall deal with the decidability part of the stated theorems. In a third one we shall deal with the techniques used for the universality part of the results. In the last part, we shall consider a new direction for founding out a new kind of frontier result about decidability and universality.

2 The results

The number of colours for a Turing machine program is a decidability criterion for Turing machines on $\{0, 1\}$. It is stated by the following theorem:

Theorem 1 ([11] and [12]). *The number of colours is a decidability criterion for Turing machines on alphabet* $\{0, 1\}$ *with* 3 *as a frontier value.*

A similar result has later been established for non-erasing Turing machines on $\{0, 1\}$, *i.e* Turing machines which can never replace 1 by 0 – symbol 0 is considered as the blank symbol of the tape:

Theorem 2 ([4]). *The number of colours is a decidability criterion for non-erasing Turing machines on alphabet* $\{0, 1\}$ *with* 5 *as a frontier value.*

The laterality number is also a criterion for Turing machines on $\{0, 1\}$ as it is now stated:

Theorem 3 ([11] and [8]). *The laterality number is a decidability criterion for Turing machines on alphabet* $\{0, 1\}$ *with* 2 *as a frontier value.*

This is also the case for non-erasing Turing machines with a different number as a frontier value:

Theorem 4 ([5] and [6]. *The laterality number is a decidability criterion for non-erasing Turing machines on alphabet* $\{0, 1\}$ *with* 3 *as a frontier value.*

In section 5 of our survey, we shall see a new direction jointly examined by Maurice Margenstern and Liudmila Pavlotskaya, generalizing the setting introduced in [14] by Liudmila Pavlotskaya. The direction consists in considering a new machine computation based on a Turing machine working with a finite automaton in the following conditions: the tape is initially restricted to the input word with the Turing machine head scanning some letter of the word and the automaton set on some state. The machine head is assumed to never go to the left of the leftmost symbol of the initial word. Each time when the Turing machine goes out from the word, the tape is extended by one cell. The content of this cell is determined by the finite automaton according to its own state and the state under which the Turing machine head exited. The following result holds:

Theorem 5 (unpublished). *The number of instructions of the Turing machine, whatever the number of states of the automaton is, is a decidability criterion for couples of Turing machines and automata associated as above indicated, with* 5 *as a frontier value.*

3 Decidability theorems

As pointed out in our introduction, the first proof of the decidability of the halting problem for 2×2 Turing machines was a tedious reduction by cases. Pavlotskaya's proof, a very short one, is based on a different idea. She introduced various integer valued *mathematical functions* connected with the behaviour of the head of the Turing machine on its tape. Technically, Pavlotskaya's proofs consist in proving that the functions she introduced in the general case, in an abstract way, turn out to be *recursive*, from which it ensues that the halting problem is decidable for the corresponding set of machines.

As indicated in our introduction, this proof remained unnoticed a long time. And so, another proof was found out by Volker Diekert and Manfred Kudlek, which takes place in a completely different setting. These authors succeeded in characterizing the *languages* constituted of the words on $\{0, 1\}$ on which the considered Turing machine halts. They proved that in almost all cases, such a set is a rationnal language, see [1] and [3]. In the remaining cases, it turns out to be a linear language. Here the proof is also a reduction by cases, strongly simplified by symmetry considerations and a general lemma based on the motion of the machine head which applies to a large number of machines. Details can be found in [3].

For what is the decidability criteria, the decidability part appears in Pavlotskaya's proof of the decidability of the point 2×2 in figure 1. Indeed, that result is an immediate corollary of the following lemma, which is also the decidability part of theorem 2.

Lemma 6 [11]. *If the program of a Turing machine T on $\{0, 1\}$ contains a single left instruction, then the halting problem is decidable for machine T.*

In the same paper, Liudmila Pavlotskaya proves the decidability part of theorem 1. Say that a Turing machine is *unilateral* if its program contains instructions with the same move except, possibly, stationnary instructions. It is clear that the halting problem is decidable for unilateral machines in $\mathcal{M}$. Combining that fact with lemma 6 reduces drastically the number of cases to be scrutinized.

3.1 A general approach

However, all the decidability parts of theorems 1 to 4 can be proved according to the same general plan clearly defined in [5] and [7].

The starting point consists in noticing that the decidability of the halting problem is obtained as soon as we get an algorithm for *recognizing* a non-halting computation. Indeed, a halting computation always eventually halts and so the algorithm is trivial: wait until the halting occurs.

Next, it can be noticed that non-halting computations, which, by definition, are computations in *infinite* time, may turn out to use either finite space or infinite space. The first case leads to the occurrence of two indentical configurations: this is a trivial case of non-halting, easy to recognize.

For conveniently analyzing the second case, we need accurate tools in order to describe the motion of the machine head on its tape. This is done in [4, 5] and mainly in [7]. In our settings, this makes use of *partial* recursive functions. We have to prove that they are *total* recursive. The corresponding proofs of [4, 5, 7] are constructive.

Let us informally summarize the notions introduced in these papers: at inital time, the *current bounds* of the configuration are the ends of the smallest segment containing both the initial word written on the tape and the machine head outside which all cells contain the blank symbol. Then, an *exit* is any time when the head goes out by one cell of the current bounds of the configuration. And for the next time, the current bound is moved to that latter position. We define *extremal exits* to be exits at which a half-turn occur, *i.e.* a change of direction in the motion.

Let us now turn back to the case when the machine motion is infinite in space. Trivial cases of unilateral motions put aside, it can easily be seen that either there are infinitely many extremal exits only on the right side, or there are infinitely many extremal exits on the other side, or there are infinitely many extremal exits on both sides. Say, respectively, right infinite case, left infinite case and traversal case.

First, consider the right infinite case. Between two consecutive extremal exits, there is a *leftmost* position for the head of the machine, we shall say an *lmp*.

This is below illustrated by figure 2.

It is now possible to formulate an important necessary and sufficient condition of ultimately periodic motion for the machine head of a Turing machine in the case it does not halt on the given data, see [4, 5, 7]:

Lemma 7 ([5]). *Let p_i be the sequence of absolute lmp's, S_i the extremal right exit which just occured before lmp time corresponding to p_i and l_i the distance*

from p_i to $v(S_i)$, position of the head at time S_i. The motion of the machine is utimately periodic if and only if:

$$\liminf_{i \to \infty} l_i < +\infty,$$

and this condition is recursively enumerable,

where an *lmp* p_i is said to be *absolute* if the head never goes to the left of that position after the time it is reached as an *lmp*.

Notice that the lim inf condition precisely states that there is an integer L and there are infinitely many i's such that $\ell_i \leq L$ for all those selected i's.

By symmetry, analogous results hold for the left infinite case.

In [7], a more accurate necessary and sufficient condition of ultimately periodic motion is given. It intuitively says that when the head goes back to the left after an extremal right exit, the next *lmp* cannot be too often too far from the previous *lmp*.

Lemma 7 appears to solve the problem for most cases belonging to both right and left infinite cases. And so, proofs focus the attention on the 'difficult' cases where the necessary and sufficient conditions of ultimately periodic motion cannot be satisfied, in particular, in the traversal case.

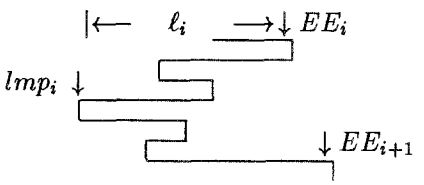


Fig. 2. On this figure, the horizontal pieces of line indicate the spaces scanned by the machine head and vertical lines indicate half-turns. This provides a condensed space-time diagram of that motion: moves in the same direction remain on the same horizontal line. EE_i is the i^{th} extremal exit (here, on the right side).

The analyses given by the decidability proofs of [4, 7] show that 'difficult' cases do happen. Up to now, what happens is an ultimately periodic increasing of the ℓ_i's or the length of traversals. It is somehow difficult to prove that such a condition is a sufficient one. Generally speaking it is surely not the single possibility of 'difficult' motion in the case of infinite space.

3.2 Using the notion of colour

The decidability part of theorem 2 uses theorem 1. It also uses a precise characterization of what could be called 'decidable' sets of five colours, see [4]. In order to state the corresponding property, we say that a colour is of *pure move*

if and only if its output symbol is the same as its input one: corresponding instructions perform their motion without altering the content of the tape cells. The following property holds:

Lemma 8 [4]. *If in the program of a non-erasing Turing machine on* $\{0,1\}$ *at least one of the pure move colour is missing, then the halting problem is decidable for that machine.*

The proof of lemma 8 is typically based on the above general considerations.

It should be noticed that lemma 8 appears not only in the proof of the decidability part of theorem 2 but also for a drastic reduction of the cases appearing in the decidability proof of theorem 4, allowing to lower the number of main cases to be examined from twenty one downto four. Moreover, with the help of particular considerations lemma 8 allows to discard three of the remaining four cases. And so, only in this last one the 'difficult' cases mentionned above do appear. See [7] for a thorough analysis.

4 Universality theorems

Various tools have been deviced for proving the universality of several computational models.

The main idea is to *simulate* a process already known to be universal. Two ways are used for this purpose. In some cases, the process is directly implemented in the considered system of computations: we shall speak of an *executive code*. In the other cases, the simulation consists in building an *interpreter* of a class of *processes* known to contain a universal element. In the simulating computation, simulated elements are *encoded* in a suitable form.

Notice that like russian dolls, simulations can be nested. This is often used for making *small* universal objects, small with respect to the size of the program.

4.1 Two register machines

The most popular tool of simulation is certainly the simulation of two register machines which are known to be universal, see for instance [10].

This is the case, in particular, for the universality parts of theorems 1 and 3. It is worth noticing that the proof of theorem 1, see [12] brings in a new ingredient which turns out to be very robust with respect to constraints.

Recall that for simulating a two register machine, it is enough to encode the content of its registers, say x and y, as a number $X = 2^x.3^y$. In this context, incrementation of one register is simulated by multiplication of X by 2, of the other one is simulated by multiplying X by 3. It is easy to see that for implementing decrementation, it is enough to simulate division by 6. But, for strong limitations on the number of colours or of laterality, it is not known how to divide by 6. Paper [12] shows that the situation is saved by simulating the following mapping: $x \mapsto x + \frac{x}{6}$, called *pseudo-division* by 6. This boils down to multiply

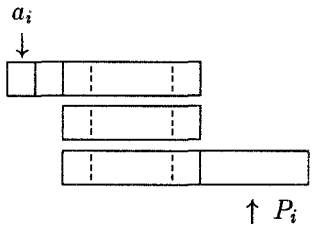
x by $\frac{7}{6}$. This introduces powers of 7 in the prime decomposition of X but this keeps the right value for the exponents of 2 and 3 in that decomposition.

4.2 Tag-systems

Another important tool, far less known than two register machines, but very powerful, are tag-systems. There are used, in particular, for making the smallest universal Turing machines: see [10, 16, 17, 18].

A p-tag-system – p a fixed positive integer – is a calculus which is associated to a mapping from alphabet A into A^*, the set of words on A. The image of $a \in A$ is its *production*. One step in the computation consists in performing the following three operations, illustrated, below on the right, in the case $p = 2$:

- a_i, first letter of the *tagged* word, *i.e.* submitted to the tag-system, is memorized ;
- first p letters of the word are erased ;
- P_i, production associated with a_i, is appended at the end of what remains from the tagged word.



Repeat the process on the word thus obtained until either the tagged word has less than p letters, or the just obtained tagged word comes from a word, the first letter of which was a *halting* letter. By definition, halting letters are distinguished letters of A making the computation halt. A single halting letter is enough, which is assumed in most simulations.

Coke-Minsky theorem (see, for instance, [10]) states that any Turing machine in $\mathcal{M}$ can be simulated by a 2-tag-system. On another hand, Wang proved that the halting problem is decidable for any 1-tag-system ([21]).

This tool is used in the universality part of theorems 2 and 4. In theorem 4, three nested simulations are used: the 218 state non-erasing machine on $\{0, 1\}$ simulates another Turing machine, say Z, on a richer alphabet that simulates a three head machine on $\{0, 1\}$ which, on its turn, simulates an interpreter of 2-tag-systems. It is enough to put the encoding of a two-register machine simulating a universal Turing machine for obtaining the universality of the non-erasing machine. Notice that the non-erasing machine simulates machine Z by always 'refreshing' the current configuration: it endlessly replicates the content of the leftmost significant cell at the other end of the configuration.

5 A new direction

In section two, we described the computation associated to a couple constituted of a Turing machine and a finite automaton.

The decidability part of theorem 5 uses the general setting we indicated in section 3 which has to be extended due to the association with a finite automaton. It also uses the notion of colour but, mainly a new notion is taken into

account for leading the splitting process into cases and for reducing the number of cases to consider. It is the notion of *phase* which is a dual notion to the notion of colour. The phase of an instruction is the projection constituted of the input state, the output state and the move of the instruction. The proof also introduces the notion of *connectivity* of a Turing machine program. Informally, a program is connected if and only if there are infinite computations which infinitely often make use of all the instructions of the machine. This allows to rule out a great number of cases. Then, inside each remaining case, colour type arguments allow to reduce again the number of subcases. Lemma 7 and its stronger version are also very useful in this proof.

The universality proofs of theorems 1 and 5 basically use simulations by a two register machine. They also use the pseudo-division introduced in [12]. But in both cases, more must be done. What is needed is a *uniform* simulation of mutiplication by 2, by 3 **and** of pseudo-division by 6. This means that the *same* algorithm is used for computing the operations, depending on a parameter. According to the value of the parameter, it is then possible to simulate now multiplication by 2, now multiplication by 3 and now, pseudo-division by 6. This allows to satisfy the constraint of two left instructions in the program for theorem 1, also see [8]. In theorem 5, a first machine is built which simulates the computation explained in [8] for proving theorem 1. It is a machine with 8 instructions, associated to a finite automaton which contains a great lot of the information needed for the computation, especially the connection between the instructions used by the simulated two register machine. Indeed, this couple works as an executive code. It is not an interpreter. Then, the couple constructed with a 5 instruction Turing machine simulates the couple with a machine with 8 instructions. Full details can be found in technical report [9].

Acknowledgements

The author is much indebt to S.I. Adian for very valuable comments. He also has to acknowledge both grant *HighTech. EV* №950525 given by NATO Division of Scientific Affairs and Metz University Institute of Technology for making possible Liudmila Pavlotskaya's visits in 1995 and 1996, thus providing the best conditions for common works with her.

References

1. Diekert V. and Kudlek M. Small Deterministic Turing Machines, Papers on Automata and Languages, Dep. of Math., Karl Marx Univ. of Economics, Budapest, 1988-4, p.77-87, (1989)
2. Hermann G.T. The uniform halting problem for generalised one state Turing machines, Proceedings of the Ninth Annual Switching and Automata Theory Symposium, (1968)
3. Kudlek M. Small Deterministic Turing Machines, Theoretical Computer Science, **168-2**, special issue on Universal Machines and Computations, 241-255, (1996)

4. Margenstern M. Turing machines: on the frontier between a decidable halting problem and universality. Lecture Notes in Computer Science, **710**, 375-385, in *Proceedings of FCT'93*, (1993)
5. Margenstern M. Non-erasing Turing machines: a new frontier between a decidable halting problem and universality. Lecture Notes in Computer Science, **911**, 386-397, in *Proceedings of LATIN*'95), (1995)
6. Margenstern M. Results on the halting problem, LMPS'95 International Conference, Florence, August 1995
7. Margenstern M. The laterality problem for non-erasing Turing machines on $\{0, 1\}$ is completely solved. RAIRO, 39p. (1997) (to appear)
8. Margenstern M. and Pavlotskaya L. Deux machines de Turing universelles à au plus deux instructions gauches. *C.R.A.S., Paris*, **320**, I, 1395-1400, (1995)
9. Margenstern M. and Pavlotskaïa L. Vers une nouvelle approche de l'universalité concernant les machines de Turing, LITP/IBP research report N°95.58 (1995)
10. Minsky M.L. Computation: Finite and Infinite Machines. Prentice Hall, Englewood Cliffs, N.J. (1967)
11. Pavlotskaya L.M. Razreshimost' problemy ostanovki dlja nekotorykh klassov mashin T'juringa. Matematicheskie Zametki, **13**, (6), 899-909, Ijun' 1973 899-909. (transl. Solvability of the halting problem for certain classes of Turing machines, Notes of the Acad. Sci. USSR, 13 (6) Nov.1973, 537-541)
12. Pavlotskaya L.M. O minimal'nom chisle razlichnykh kodov vershin v grafe universal'noj mashiny T'juringa. Diskretnyj analiz, Sbornik trudov instituta matematiki SO AN SSSR, **27**, 52-60, (1975) (On the minimal number of distinct codes for the vertices of the graph of a universal Turing machine) (in Russian)
13. Pavlotskaya L.M. Dostatochnye uslovija razreshimosti problemy ostanovki dlja mashin T'juring, Avtomaty i mashiny, 91-118, 1978 (Sufficient conditions for halting problem decidability of Turing machines) (in Russian)
14. Pavlotskaya L.M. On machines, universal by extensions, Theoretical Computer Science, **168-2**, special issue on Universal Machines and Computations, pp.257-266, (1996)
15. Priese L. Towards a precise characterization of the complexity of universal and nonuniversal Turing machines, SIAM Journal of Computation, **8**, 4, 508-523, (1979)
16. Rogozhin Ju.V. Sem' universal'nykh mashin T'juringa. Matematicheskie Issledovanija, **69**, 76-90, 1982 (Seven universal Turing machines) (in Russian)
17. Rogozhin Ju.V. Universal'naja mashina T'juringa s 10 sostojanijami i 3 simvolami. Matematicheskie Issledovanija, **69**, 76-90, 1992 (A universal Turing machine with 10 states and 3 symbols) (in Russian)
18. Rogozhin Ju.V. Small universal Turing machines, Theoretical Computer Science, **168-2**, special issue on Universal Machines and Computations, 215-240, (1996)
19. Shannon C.E. A universal Turing machine with two internal states. Ann. of Math. Studies, **34**, 157-165, (1956)
20. Turing A.M. On computable real numbers, with an application to the Entscheidungsproblem. Proc. Lond. Math. Soc., ser. 2, **42**, 230-265 (1936)
21. Wang H. Tag Systems and Lag Systems, Mat. Annalen, 152, (1963), 65-74