ELSEVIER

Theoretical Computer Science 168 (1996) 257–266

Theoretical Computer Science

On machines, universal by extensions

Liudmila Pavlotskaya*

Moscow Power Engineering Institute, Moscow, Russia

Abstract

A new definition of universal Turing machines is introduced which allows to obtain universal Turing machines with a very small number of program instructions. Here such a program with only seventeen instructions is supplied.

1. Introduction

According to the definition first formulated by Davis [1], a Turing machine is universal if the set of its configurations leading to a completed computation is m-complete. The law of succession between configurations in the computing process which is assumed by this definition reflects our intuitive representation of algorithms applied to a set of finite objects. In particular, it is assumed that each time the machine head goes out of a word, it always reads the same symbol, called blank, in a cell which it did not previously visit. But, in order to simulate any kind of computations, a universal machine may use a non-recursive set of "codes" and, in that case, the machine cannot itself define the end of the word which is the code of a certain computation. This means that the notion of "word end" for the machine itself loses its meaning. However, if it is demanded that the set of codes should be recursive, then the definition of universal machine would become unapplicable.

It seems to us that the following property of universality does not correspond to the intuitive representation of machines on which it is possible to simulate the computation of any other machine. In general, it is impossible to restore the result of the simulated computation from the result obtained by the universal machine. If the simulated machine computes function $f(n)$, then, in order to obtain this value by the universal machine, it is necessary to make two copies of that machine run $f(n)$ times, each time finding new initial configurations. It seems that our informal representation of universal machine does not completely correspond to the formal one.

* E-mail: vm@mpei-rt.msk.su.

PII S0304-3975(96)00079-5

Putting aside the question of which of the below-considered machines could be considered as universal from the intuitive point of view, we define below Turing machines which are universal on certain sets of extensions and we give such an example of a machine, the program of which contains 17 instructions and which is universal on periodic extensions.

In [7, 8], an example of a universal machine is given with a program containing 23 instructions. No universal machine in the traditional meaning with a shorter program has yet been known.

2. Basic definitions

We consider Turing machines, the tape of which is infinite in one direction, from left to right, with $X = \{x_1, x_2, \ldots, x_m\}$ as the alphabet of input symbols, $Q = \{q_1, q_2, \ldots, q_l\}$ as the set of states of the machine head and with a program, containing instructions of the form $q_i x_j M x_k q_l$, where M is the symbol of move to the right (R) or to the left (L). The execution of such a command consists in replacing in the scanned cell symbol x_j by symbol x_k, setting the head in state q_l instead of q_i, and moving along the tape by one cell to the right if $M = R$, and by one cell to the left if $M = L$. If the head is in state q_i and reads x_j, and if the program has no instruction beginning with $q_i x_j$, then the computation halts.

Let the tape cells be numbered $1, 2, \ldots$ The piece of the tape constituted of cells numbered $i, i+1, \ldots, j$ is denoted $[i, j]$. Consider that at the initial time, word x is written on $[1, |x|]$, where $|x|$ is the length of word x, and that the head reads one of the corresponding cells. By word written on the tape at a given step of computation, we understand the initial word or the word obtained by recording on the initial one the symbols contained in all the cells on the right hand of $[1, |x|]$ that were visited by the head up to the considered time.

Call machine configuration the set consisting of word $x = x_{i_1} \ldots x_{i_r}$ written on the tape, of the state of the head, say q_i and of the number of the scanned cell. The set of these data can be written as a single word $x_{i_1} \ldots q_i x_{i_k} \ldots x_{i_r}$, which we also call configuration.

If during the computation starting from configuration K, the head does not go out of the ends of that part of the tape which contains word x, we shall consider that the computation on K is defined and, in that case, it can be finite or infinite. In the other case, the computation on K is not defined and call *extremal configuration* word $x'q_j$ if at that moment of exiting out of the word end, word x' is written on the part $[1, |x|]$ and the head is under state q_j.

Definition 1. Call word p finite extension of extremal configuration xq_i if the computation on configuration xq_ip is defined and finite. Call also word p finite extension for those configurations such that the computation starting from them leads to configuration xq_i.

3. Machines which are universal on a set of configurations with extensions belonging to a given set

Let $\mathscr{K}$ be the set of configurations of machine F and let $\mathscr{K}_{\text{fin}}(\mathscr{P})$ be the set of those configurations of $\mathscr{K}$ for which either the computation is defined and finite or there are finite extensions belonging to set $\mathscr{P}$.

Assume that a numbering is put on the set of all configurations.

Definition 2. Call a Turing machine universal on the set $\mathscr{K}$ of configurations with extensions in $\mathscr{P}$ if $\mathscr{K}_{\text{fin}}(\mathscr{P})$ is m-complete.

Assume that F is universal in our sense. How F can be used in order to simulate machine Z computing function $f_Z(n)$ in the natural encoding? By definition, it follows that there is a recursive function $g(n)$ such that $f_Z(n)$ is defined if and only if $g(n)$ is the number of a configuration from $\mathscr{K}_{\text{fin}}(\mathscr{P})$.

Let n_0 be an integer. Let us make F start its computation from configuration K_0 with number $g(n_0)$. If the computation on K_0 is defined and finite then function $f_Z(n)$ is defined for $n = n_0$. When the computation is defined an infinite, function $f_Z(n)$ is not defined for $n = n_0$. In the case when the computation on K_0 is not defined the machine reaches a certain extremal configuration xq_i. Let us now make F compute on configuration xq_ip_1 where $p_1 \in \mathscr{P}$. If the new computation is defined and finite, word p_1 is a finite extension of configuration xq_i and in this case function $f_Z(n)$ is defined for $n = n_0$. In the case when the new computation is defined and infinite or it is not defined, let us start the computation from configuration xq_ip_2 with $p_2 \in \mathscr{P}$ and let us find out if p_2 is a finite extension of configuration xq_i. If $\mathscr{P}$ is recursively enumerable and $f_Z(n_0)$ is defined, this process will come to an end and a finite extension of configuration xq_i will be found.

Consequently, when $\mathscr{P}$ is a set of words on an alphabet with more than one symbol, it is difficult to use a machine which is universal on configurations with extensions in $\mathscr{P}$ in order to simulate arbitrary computations. For this reason, it is interesting to consider machines which are universal on extension sets with a structure easily allowing to find finite extensions or to get convinced that there are no extensions.

4. Existence of a machine with a decidable halting problem and universal in the new meaning

It is known that there is a Turing machine with $\{0, 1\}$ as input alphabet and $q_1, \ldots, q_s$ as states which is universal in the traditional meaning on the set of configurations of the form $q_1 1^n$. Let F be one of these machines and let F computations halt only when the machine head is in state q_j and it sees symbol x (with x being either 0 or 1). Then the set F_{fin} of configurations of the indicated type on which the computation halts is m-complete.

Let us transform F into the new machine G with $\{0,1,2\}$ as input alphabet and let us append the following instructions to the program of F:

$$q_i2R2\alpha,\ \alpha2R2\alpha,\ \alpha0R0\alpha,\ \alpha1R1\alpha \quad (i=1,\ldots,s)$$

$$q_jxRx\beta,\ \beta0R0\beta,\ \beta1R1\beta$$

In the program of G, instructions beginning with $\beta2$ are missing. It clearly follows from that how to transform the program of F which contains halting instructions different from q_jx in the program of G such that the latter should contain a "halting" instruction with "$\beta2$".

The set of all configurations of machine G of the form q_11^n for which there are finite extensions belonging to $\{0^k2\}$ coincide with F_{fin}. Consequently, this set is m-complete and machine G is universal on $\mathscr{K}=\{q_11^n\}$ with extensions in $\mathscr{P}=\{0^k2\}$.

The halting problem for machine G is decidable on $\mathscr{K}$, not depending on which symbol among 0 and 2 is taken as the blank symbol. Indeed, machine G does not halt when, starting on configuration q_11^n, it will always encounter symbol 0 on cells $n+1$, $n+2,\ldots$ On the other hand, when all these cells contain symbol 2, the machine, going out of the word, will halt if it is in state β and will move for ever to the right if it went out of the word in another state.

Accordingly, the following theorem holds:

Theorem. *There are machines which are universal on a set $\mathscr{K}$ of configurations with extensions in a certain set $\mathscr{P}$ for which the halting problem on $\mathscr{K}$ is decidable.*

5. Universal machines on computable extensions

Let us consider computations of machine G with extensions $p=0^k$. If the computation leads to a new extremal configuration, it is enough next to consider 0^{k+1} as a finite extension if the machine head went out of the word in state q_i, $i=1,\ldots,s$, and 0^k2 if the head went out in state β. This means that there is an algorithm, a single one for all configurations, with which it is possible to organize the determined computations with G which will be finite only on those configurations for which there are finite extensions. Under these considerations, the algorithm computing the next symbol in the possible finite extensions can be provided by a finite automaton with $Q=\{q_1,\ldots,q_s,\beta\}$ as input alphabet and $\{0,2\}$ as output alphabet.

Machines of this kind will be called universal machines on computable or determined extensions.

In the general case, assume that extensions of all computations of F on $\mathscr{K}$ leading to extremal configurations are uniquely defined by algorithm A, i.e. for each extremal configuration xq_i algorithm A defines symbol x_j (depending in the general case on x and q_i) for defining configuration xq_ix_j starting from which the computation is to be continued. Let us denote by $\mathscr{K}_{\text{fin}}(A)$ the set of configurations of $\mathscr{K}$ on which the computation halts.

Definition 3. A Turing machine is universal on computations with extensions by algorithm A if $\mathscr{K}_{\text{fin}}(A)$ is m-complete.

If it is allowed to take as A an algorithm of any complexity, the machine with the following program, $q_1 1 R 1 q_1$, $q_1 0 R 0 q_2$, $q_2 0 R 0 q_2$, is universal on configurations $\{q_1 1^n\}$ with extensions $\{0^n 1\}$ determined by a universal machine. Notice that such a machine is not universal on any fixed set of extensions, i.e. in the meaning of Definition 2.

6. Turing machines, universal on automaton extensions

There are machines for which computable extensions can be supplied by a finite automaton. Let us call such Turing machines *machines with automaton extensions.*

Let A be a finite automaton with states $S = \{s_1, \ldots, s_r\}$, with $Q = \{q_1, \ldots, q_\ell\}$ as input alphabet and $X = \{x_1, \ldots, x_m\}$ as output alphabet. Its working is given by two functions $p : S \times Q \to X$ and $d : S \times Q \to S$.

Deterministic computations on Turing machines using automata run as follows.

Let the machine reach extremal configuration xq_i and let the automaton be in state s_j at this time of the computation. Symbol $p(s_j, q_i)$ is then written in the cell on which the machine head went out, and the automaton goes to state $d(s_j, q_i)$. Later, the machine performs its computation starting from configuration $xq_i p(s_j, q_i)$ and the automaton stays in state $d(s_j, q_i)$ until the machine reaches the next extremal configuration. At initial time, the automaton is in a distinguished state called initial state.

Machines which are universal on extensions computed by using this algorithm are called *universal Turing machines on automaton extensions.*

An example of a machine which is universal on periodic extensions, hence, on automaton extensions, is given below.

The universal Turing machines with a single left-hand instruction obtained in [2] compute consecutive iterations of a function belonging to a special kind. Denote $T(n)$ such a function. Let number $d \geqslant 2$ be fixed and let functions $p(r)$ and $q(r)$ be given for $0 \leqslant r < d$, with $p(r) < d$. Starting from $n = n_0$, a sequence of numbers, say n_k, is obtained. For each n_k let m_k and r_k respectively denote the quotient and the remainder in the division of n_k by d. Let us then define n_{k+1} as follows:

$$n_{k+1} = n_k - d + p(r_k) \quad \text{if } m_k \neq 0,$$

$$n_{k+1} = k \cdot d + r_k + q(r_k) \quad \text{if } m_k = 0.$$

When $m_k = 0$, $T(n)$ is computed and then $T(n) = n_{k+1}$.

The process of computing iterations of T halts if remainder r_0 takes one of the values of some finite set i.e. $r_0 \in \{r_{10}, \ldots, r_{l0}\}$.

It follows from [2, 3] that for any partial recursive function $f(n)$ there is such a function $T(n)$ that we have: the process of computing the iterations of $T(n)$ for numbers of the kind $C*2^{2^n}+2$ halts if and only if $f(n)$ is defined and in this case the result of the

computation is $C*(2^{2^{f(n)}}*7^{g(n)}+1)$. Here $C = 2^N$, where N is the number of instructions in Minsky's machine used to compute function $f(n)$.

If he/she wants to know the details of the proof of the latter result, the reader can be recommended papers [2, 3]. Here is only outlined the basic idea for obtaining function $T(n)$.

It is known that for any partial recursive function $f(n)$ there is a Minsky machine transforming 2^{2^n} into $2^{2^{f(n)}}$, the program of which contains instructions of the following types: $\times 2$, $\times 3$, $/6,a$, halt instruction, see for instance [5]. Executing instructions of the first two types consists in multiplying the given number by 2 or 3, respectively, and turning to the execution of the next instruction. Executing instructions of the third type means to divide the given number by 6 and the execute instruction number a if the number is divisible by 6. If the number is not divisible by 6, then the machine executes the next instruction.

Let us change each instruction of type three in this algorithm into instruction $+n/6,a$, the execution of which has the same sense with the only difference that if the given number is divisible by 6 it is multiplyed by $\frac{7}{6}$. This algorithm transforms 2^{2^n} into $2^{2^{f(n)}} * 7^{g(n)}$ where $g(n)$ is the number of multiplications by $\frac{7}{6}$ occurring during the computation process.

Let the instructions of this algorithm be numbered $1, 2, \ldots, N$, where N is the number of the halt instruction. Then any step of the computation consists in executing one instruction. Let us assume that instruction **i**, a $\times 2$-instruction, is applied to number n. It can be managed that: $T(2^N * n + 2^i) = 2^N * 2n + 2^{i+1}$ if $d = 2^N * 42$, and for $r = 2^N * t + 2^i$, $p(r) = 2^N * 21$, $q(r) = 2^N * t + 2(i+1) - 2^i$.

By analogy it is possible to replace the execution of the instructions of other types by computing function $T(n)$, if the following features are taken into consideration:

$$T(n) = 2n \text{ for } d = 42, \quad p(r) = 21, \quad q(r) = r;$$

$$T(n) = 3n \text{ for } d = 42, \quad p(r) = 28, \quad q(r) = 2r;$$

$$T(n) = \tfrac{7}{6} \cdot n \text{ for } d = 42, \quad p(r) = 6, \quad q(r) = \frac{r}{6}.$$

The machine, the program of which is given below, computes the iterations of function $T(n)$ under some encoding with the condition that the extension of initial configurations should belong to the set of $\{p^n\}$, where p is a fixed word on the machine alphabet.

Here is the program of machine T:

$$\begin{array}{lll} \alpha 0 R 5 \alpha & \alpha 6 R 6 \gamma & \gamma 2 R 5 \beta \\ \alpha 1 R 4 \beta & \beta 0 R 5 \beta & \gamma 3 R 5 \gamma \\ \alpha 2 L 0 \alpha & \beta 1 R 4 \beta & \delta 0 L 0 \delta \end{array}$$

$\alpha 3R5\alpha$ $\beta 3R5\delta$ $\delta 2L2\alpha$

$\alpha 4L1\alpha$ $\gamma 0R5\gamma$ $\delta 3R1\delta$

$\alpha 5L0\alpha$ $\gamma 1R1\gamma$

Word p is of the form

$$p = \mathrm{cod}(0)\ldots\mathrm{cod}(r)\ldots\mathrm{cod}(d-1),$$

where $\mathrm{cod}(r) = 2$ if $r \in \{r_{10},\ldots,r_{l0}\}$ and, if $r \notin \{r_{10},\ldots,r_{l0}\}$,

$$\mathrm{cod}(r) = 30^{p(r)}2 \quad \text{for } r + p(r) < d,$$

$$\mathrm{cod}(r) = 330^{p(r)}2 \quad \text{for } r + p(r) \geqslant d.$$

Here still is used another property of the functions whose iterations are computed by the machines built in [2]. These functions are such that if remainder r is obtained during the computation, then $r+1$ never appears afterwards as a remainder. This is the reason why after $\mathrm{cod}(r)$, it is possible to put $\mathrm{cod}(r+1)$ in word p, where the latter cod is defined by

$$\mathrm{cod}(r+1) = 3^{t+1}0^{q(r)-1}2,$$

where t is the number defined by

$$t \cdot d \leqslant r + q(r) < (t+2) \cdot d.$$

Machine T simulates the computation of function $T(n)$ by transforming the configuration attached to n_k by the one which is attached to n_{k+1}. Each configuration attached to a given n_k has the form:

(I) $611\ldots144\ldots4\underset{*}{3}(3)\ 00\ldots0\ 2\ \mathrm{cod}[r_k+1]\ldots\mathrm{cod}(d-1)pp\ldots$

where $*$ indicates the machine head position.

This represents the encoding of n_k in the following meaning:

- the number of 1's is m_k,
- the number of 4's is $k+1$,
- the head is in state β and is scanning the first symbol in $\mathrm{cod}(r_k)$.

Computations shall be considered on configurations with extensions in set $\{p^n\}$. In that case, finite extensions are beginnings of a periodic sequence and these extensions can be found if it is assumed that the tape cells on the right hand of the initial configuration do contain that sequence starting from the beginning.

Notice that symbol 5 plays a special role during the computation. If the head had replaced symbol x by 5 in a cell, the later computation does not depend on what happens in that cell. In that case, say that the head has erased symbol x. Indeed, the head may write symbol 5 in the scanned cell, only while moving to the right. Consequently, it may come back to this place only in state α since all left instructions lead to state α. When this happens, the head performs instruction $\alpha 5L0\alpha$ and

it may later come on this 0 being in state α, β or γ. In all these cases, it writes again symbol 5 and remains in the same state. For this reason, we shall not write symbols 5 in the representations of configurations when doing this makes things more simple. Notice that these symbols may be placed only on the left side of the head. In the same way, say that the head erases symbol 2 when it performs instruction $\alpha 2L0\alpha$.

Consider now machine T computation starting from configuration (I). The head first erases the first 3, then replaces the possible second one by one while performing instructions $\delta 0R0\delta$ and $\delta 2L2\alpha$. Configuration (I) is now transformed into the following:

(II) $611\ldots144\ldots4\ (1)\ 00\ldots\underset{*}{0}\ 2\ \mathrm{cod}[r_k+1]\ldots\mathrm{cod}(d-1)pp\ldots$

Starting from this time, the head remains in state α and erases one 0. It goes then rightward, erasing all symbols until it meets the first 2, also erasing this symbol. Then it moves leftward until it reaches the first occurring 2, erases that symbol and then erases all symbols on the right side until it reaches the next symbol 2, repeating this cycle until the configuration looks like the following:

(III) $611\ldots144\ldots4\ (1)\ 55\ldots\underset{*}{5}\ 0\ \mathrm{cod}[r_{k+1}]\ldots\mathrm{cod}(d-1)pp\ldots$

The head erases the $p(k)$ 0's which lie on its right, also counting the 0 it is scanning in configuration (II) under state α. Starting from this time it goes to the right, looking after the next 1, then replacing it with 4, turning to state β and, remaining under that state, moving to the right until it reaches the first occurring 3 in the encoding of remainder r_{k+1}.

Configuration (III) is replaced by

(IV) $611\ldots144\ldots4\ (4)\ \underset{*}{3}(3)\ 00\ldots02\ \mathrm{cod}[r_{k+1}+1]\ldots$

The number of 4's in (IV) is by 1 greater than that number in configuration (I). The number of 1's is unchanged when, in configuration (I), $\mathrm{cod}[r_k]$ contains two symbols 3, which matches condition $m_{k+1}=m_k$ when $r_k+p[r_k]\geqslant d$. When the latter condition does not hold and, consequently, $\mathrm{cod}[r_k]$ contains a single 3, the number of 1's in (IV) is equal to m_k-1.

When the machine reaches configuration (III) with the form

$$6444\ldots455\ldots\underset{*}{5}0\ \mathrm{cod}[r_{k+1}]\ldots$$

its head is under state α, and it moves to the left until it reaches the first cell, containing symbol 6. At this point, it turns to state γ and, remaining in that state, it reaches $\mathrm{cod}[r_k]$, erasing that word and turns to state β as it reads the first symbol of $\mathrm{cod}[r_{k+1}+1]$ which has the form

$$\underset{*}{3}\ 3^t0^{q[r_{k+1}]-1}$$

Now, the word lying to the left of the head reads as 61^{k+1}, with 5's being skipped. Starting from that time, the machine works as it did while beginning the computation on configuration (I): after erasing all 0's contained in that encoding, it finds the nearset 1 on its left, changes it into 4 and moves to the beginning of the encoding, which is the starting configuration for computing the next iteration of function $T(n)$:

$$6\ 1^{k+1}\ 1^{t-1}\ 4\ \underset{*}{3}(3)00\ldots02\ \mathrm{cod}[r+1]\ldots\mathrm{cod}(d-1)pp\ldots$$

The head is in state β, and r is the remainder in the division of $r_{k+1}+q[r_{k+1}]$ by d. When r belongs to the set of remainders which indicates that the computation is completed, $\mathrm{cod}(r)$ is 2 and the head halts since there is no instruction with $\beta 2$ as input couple and in this case the result of the computations is $(k+t)d+r$.

7. Conclusion

As previously indicated, we give here the construction of a very simple machine for a special case of automaton extensions, namely, periodic ones. The reader may wonder what happens with more complex automaton extensions. Is it possible to get more simple machines with the counterpart of a more complex automaton? The answer to this question is yes. The author is preparing with Maurice Margenstern a paper which gives more simple Turing machines which are universal on automaton extensions. In the technical report [4], two such machines are constructed: one with 8 instructions and another with 5 instructions. On the other hand, in a paper in preparation with the same co-author, it will be proved that machines with 4 instructions can never fail to have a decidable halting problem on automaton extensions.

Acknowledgements

High Tech. EV No 950525 grant given by NATO Scientific Affairs Division has provided the best conditions for writing this paper as well as the technical report [4] in collaboration with Maurice Margenstern.

References

[1] M.D. Davis, A note on universal Turing machines, in: C. Shannon and J. McCarthy, eds., Automata Studies (Princeton University Press, Princeton, NJ, 1956) 167–175.

[2] M. Margenstern and L. Pavlotskaya, Deux machines de Turing universelles à au plus deux instructions gauches, *Comptes Rendus de l'Académie des Sciences (Paris)* **320** (I) (1995) 1395–1400.

[3] M. Margenstern and L. Pavlotskaya, Deux machines de Turing universelles: l'une sur $\{0,1\}$ avec deux instructions gauches, l'autre sur $\{0,1,2\}$ avec une seule instruction gauche, LITP Research Report No. 95.25, Institut Blaise Pascal, 1995.

[4] M. Margenstern and L. Pavlotskaya, Vers une nouvelle approche de l'universalité concernant les machines de Turing, LITP Research Report No. 95.58, Institut Blaise Pascal, 1995.

[5] M.L. Minsky, *Computation: Finite and Infinite Machines* (Prentice-Hall, Englewood Cliffs, NJ, 1967).
[6] L.M. Pavlotskaya, Sur les systèmes de calcul, *Actes de MCU'95/UMC'95 Proc.* (1996) 168–177, to appear.
[7] Iu.V. Rogozhin, Sem' universal'nikh machin T'juringa, *Matematicheskie Issledovanija*, 69, Kichinev, Moldova (1982) 76–90.
[8] Iu.V. Rogozhin, Small universal Turing machines, this TCS special issue.