ELSEVIER

Theoretical Computer Science 289 (2002) 573–590

Theoretical Computer Science

www.elsevier.com/locate/tcs

On the presence of periodic configurations in Turing machines and in counter machines

Vincent D. Blondel[a,*], Julien Cassaigne[b], Codrin Nichitiu[c]

[a] *Division of Applied Mathematics, Center Cesame, Université catholique de Louvain, 4 avenue Georges Lemaitre, B-1348 Louvain-la-Neuve, Belgium*

[b] *Institut de Mathématiques de Luminy—CNRS, 163 avenue de Luminy, Case 907, 13288 Marseille Cedex 9, France*

[c] *EURISE, Université Jean Monnet St Étienne, 23, rue du Dr Paul Michelon, 42023 St Étienne Cedex 2, France*

Received September 2000; received in revised form September 2001; accepted September 2001
Communicated by E. Goles

Abstract

A configuration of a Turing machine is given by a tape content together with a particular state of the machine. Petr Kůrka has conjectured that every Turing machine—when seen as a dynamical system on the space of its configurations—has at least one periodic orbit. In this paper, we provide an explicit counterexample to this conjecture. We also consider counter machines and prove that, in this case, the problem of determining if a given machine has a periodic orbit in configuration space is undecidable.

1. Introduction

A *Turing machine* is an abstract deterministic computer with a finite set Q of *internal states*. The machine operates on a doubly infinite tape of cells indexed by an integer $i \in \mathbf{Z}$. Symbols taken from a finite alphabet Σ are written on every cell; a *tape content* can thus be seen as an element of $\Sigma^{\mathbf{Z}}$.

At every discrete time step, the Turing machine scans the cell indexed by 0, and depending upon its internal state and the scanned symbol, the machine either has no

* Corresponding author. Fax: +32-10-47-2180.

E-mail addresses: blondel@inma.ucl.ac.be (V.D. Blondel), cassaigne@iml.univ-mrs.fr (J. Cassaigne), codrin.nichitiu@univ-st-etienne.fr (C. Nichitiu).

0304-3975/02/$ - see front matter
PII: S0304-3975(01)00382-6

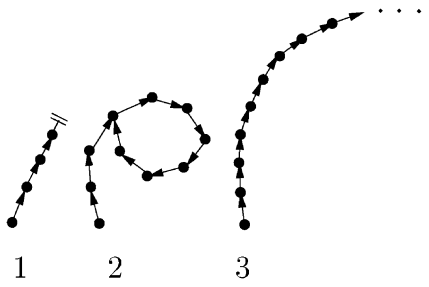

Fig. 1. The three possible types of configurations: (1) halting, (2) eventually periodic, and (3) wandering.

corresponding action or performs one or more of the following operations: replace the scanned symbol with a new symbol, focus attention on an adjacent square by shifting the tape by one unit, and transfer to a new state. A Turing machine M can thus be given by its (possibly partial) *transition function* $\delta_M : Q \times \Sigma \to Q \times \Sigma \times \{-1, 0, 1\}$. A tape content together with an internal state constitute a *configuration* of the machine. A Turing machine thus defines a (partial when δ_M is partial) function $f : Q \times \Sigma^{\mathbf{Z}} \to Q \times \Sigma^{\mathbf{Z}}$ on its configuration space $C = Q \times \Sigma^{\mathbf{Z}}$.

This is actually one of several possible models of the Turing machine (TM), namely, the Turing machine with moving tape. Another model can be obtained by making the head scan a cell of any index (the tape being immobile), and by encoding the head position (this very index) as well in the configuration. This is a Turing machine with moving head, and the space of configurations has a different topology. We will get back to this issue in a few paragraphs.

We look at computing machines as dynamical systems on configuration space and look at the possible types of trajectories. This approach contrasts with the computability one, where the configurations usually are finite objects (the considered tape content is finite, the rest of the tape being filled with "blanks") and where the process is started from precise initial configurations. Let f be the function defined by some Turing machine M on its configuration space and let c be some configuration of M. The configuration $c' = f(c)$ is the *successor* of c. If f is not defined on c, then c has no successor and is said to be *terminal*. Denote by $f^t : C \to C$ the tth iteration of f. A configuration c is *halting* (or *eventually terminal*) if $f^t(c)$ is terminal for some $t \geqslant 0$, it is *periodic* if $f^t(c) = c$ for some $t \geqslant 1$, and it is *eventually periodic* if $f^t(c)$ is periodic for some $t \geqslant 0$. Configurations that are not halting nor eventually periodic are said to be *wandering*. Thus, a configuration is either halting (1), eventually periodic (2), or wandering (3); see Fig. 1.

It is clear that a machine has a periodic configuration if and only if it has an eventually periodic configuration, and this condition is again equivalent to that of the existence of a configuration that leads to a periodic sequence of tape contents. Little is known about the possible combinations of configuration types defined by Turing machines. For the Turing machine model with a moving head as defined above, it is easy to construct machines whose configurations are all wandering. For example, the machine writing $0 \ldots 01$, with an increasing number of 0, has only wandering

configurations in the moving head model. No such simple construction seems possible for Turing machines with moving tape, and Petr Kůrka has conjectured the following in [5] (see also [7] for a discussion of this conjecture):

Conjecture (Kůrka, 1997). *A Turing machine with moving tape that has no halting configuration has a periodic configuration.*

In this paper, we analyze Kůrka's conjecture and questions related to it. We first consider counter machines rather than Turing machines. An n-counter machine with state set Q can be seen as a dynamical system on the configuration space $C = Q \times \mathbf{N}^n$. The 1-counter machine that keeps incrementing its unique counter has no periodic configuration and therefore constitutes an easy counter-example to a statement analogous to Kůrka's conjecture for counter machines. In Section 2, we prove the stronger result that, in the case of counter machines, the problem of determining if a given machine that has no halting configuration has a periodic configuration is undecidable.

In Section 3, we consider the Turing machine model with moving tape. We first provide an explicit construction of a Turing machine that has no halting configuration nor periodic configuration, thus disproving Kůrka's conjecture. The machine we construct has 36 states and operates on an alphabet of four letters. It can then be transformed into a machine with only three states. This bound on the number of states cannot be further improved since we show that machines with only two states always have periodic configurations. We also give another explicit counterexample with only six states and four letters.

2. Periodic configurations for counter machines

An n-counter machine is an abstract deterministic computing machine with a finite set Q of internal states and a finite number of registers $R_1, \ldots, R_n$ containing non-negative integers. The register values together with the internal state of the machine constitute a *configuration* of the machine. The configuration space of counter machines is thus given by $C = \mathbf{N}^n \times Q$. Depending upon its internal state and whether the registers are equal to 0, a machine can perform one of the following operations: leave the registers unchanged, increase some register R_j by 1, or decrease some register R_j by 1 (assuming $R_j \neq 0$), and move to a new internal state k. Let $\mathscr{I} = Q \times \{1, 2, \ldots, n\} \times \{-1, 0, +1\}$; the transition function of the counter machine is then defined as $\delta : Q \times \{0,1\}^n \to \mathscr{I}$, with the indicators of registers which are equal to zero belonging to $\{0,1\}^n$ (0 means the corresponding register is non-zero, and 1 means the corresponding register is indeed null), and the actions to perform belonging to $\mathscr{I}$.

This definition of a counter machine is slightly different from that given in [4], but is easily seen to be equivalent in terms of computational power. A counter machine defines a (partial when δ is partial) function $f : C \to C$ on its configuration space, in the same way a Turing Machine does (see Section 1 and Fig. 1), and the same definitions for configurations (successor, terminal, halting, wandering, eventually periodic) apply.

As explained in the Introduction, it is easy to construct counter machines that only have wandering configurations. Consider for example the 1-counter machine with one

state q_1, that keeps incrementing its unique register. This machine is defined by

$$\delta(q_1,1)=(q_1,+1,1) \text{ and } \delta(q_1,0)=(q_1,+1,1).$$

This machine has no halting nor periodic configuration; all its configurations are wandering. By adapting the proof of Theorem 1 in [2], we prove that distinguishing the counter machines that have a periodic configuration from those that have not cannot be done algorithmically.

Theorem 1. *Let M be a counter machine that has no halting configuration. The problem of determining if M has a periodic configuration is undecidable. This problem is undecidable even in the case of 2-counter machines, but is decidable for 1-counter machines.*

Proof. The proof is by reduction from the classical halting problem for counter machines; see [4]. Consider a counter machine M with m internal states labeled $q_1, q_2, \ldots, q_m$, n registers $R_1, \ldots, R_n$, transition function δ_M, and let $s=(r_1, r_2, \ldots, r_n, q_l)$ be a given configuration of M.

To establish the first part of the result we describe how to effectively construct a counter machine M' that has no halting configuration, that has $n+2$ registers $R_1, \ldots, R_n$, $V=R_{n+1}$, $W=R_{n+2}$, and that has a periodic configuration if and only if M halts on s.

The machine M' has a special state denoted by q_0. Each time when it enters the state q_0, M' executes a sequence of transitions whose effect is to store r_i in R_i, for all i, $1 \leqslant i \leqslant n$, $2\max(1,V)$ in W and 0 in V. After having done this, the machine moves into state q_* and from there moves into state q_l. (The intermediate state q_* is only introduced to facilitate the exposition of the proof.)

Then the machine starts a simulation of the machine M. The simulation is such that, before performing any of the transitions of M, the machine first increases the value of the register V by 1, decreases that of W by 1 and performs the transition of the machine M only if the value of W is not equal to 0. If the value of W is equal to 0 it returns to the special state q_0.

Thus, each state q_i of M is replaced with three states q_i, q_i', q_i'' of M', with the following transitions, for all $(b_1, b_2, \ldots, b_{n+2}) \in \{0,1\}^{n+2}$:

$$\delta_{M'}(q_i, b_1, b_2, \ldots, b_{n+2}) = (q_i', n+1, +1),$$

$$\delta_{M'}(q_i', b_1, b_2, \ldots, b_{n+1}, 0) = (q_i'', n+2, -1),$$

$$\delta_{M'}(q_i', b_1, b_2, \ldots, b_{n+1}, 1) = (q_0, 1, 0).$$

If $\delta_M(q_i, b_1, b_2, \ldots, b_n) = (q_k, j, D)$ is a transition of M, then it is changed into four transitions of M':

$$\delta_{M'}(q_i'', b_1, b_2, \ldots, n_n, b_{n+1}^*, b_{n+2}^*) = (q_k, j, D),$$

where b^*_{n+1} and b^*_{n+2} range over all four possible combinations, that is $b^*_{n+1}, b^*_{n+2} \in \{0,1\}$. We complete the construction of M' by extending $\delta_{M'}$ for each terminal configuration $(q_i, b_1, b_2, \ldots, b_n)$ of M as follows:

$$\delta_{M'}(q''_i, b_1, b_2, \ldots, n_n, b^*_{n+1}, b^*_{n+2}) = (q_0, 1, 0),$$

that is, making M' move to q_0 from all terminal configurations of M.

The machine M' we have constructed has no halting configuration. We claim that it has a periodic configuration if and only if M halts on s.

In order to prove our claim, assume first that M halts on s and let k be the number of steps after which it halts. Consider the machine M' at configuration $c = (r_1, r_2, \ldots, r_n, 0, 2\max(k,1), q_*)$. Before every simulation step of M, the value of the register V is increased by 1 and that of W is decreased by 1. After k such steps, the value of the register V is equal to k and the machine M' jumps to q_0. From there it is easily verified that M' returns to c, and so c is a periodic configuration.

For the reverse implication, assume that M' has a periodic configuration. We first observe that all trajectories in configuration space pass infinitely many times through configurations of the form $(r_1, r_2, \ldots, r_n, 0, 2k, q_*)$ for some $k \geqslant 1$. Indeed, the register W is regularly decremented when executing transitions of M'. It is therefore clear that, whatever configuration the machine M' starts from, either the machine M' reaches a terminal configuration of M, or W reaches 0 after finitely many steps. In both cases, M' then jumps to q_0, executes a sequence of transitions whose effect is to store r_i in R_i, $2\max(1, V)$ in W, and 0 in V and finally moves to q_*, thus leading to the configuration $(r_1, r_2, \ldots, r_n, 0, 2k, q_*)$ for some $k \geqslant 1$.

From this observation we conclude that if M' has a periodic configuration, then it must have one of the form $c = (r_1, r_2, \ldots, r_n, 0, 2k, q_*)$ for some $k \geqslant 1$. But a configuration of this type can only be periodic if the machine M halts on s after k steps. Indeed, if M does not halt on s, then the value of $2V + W$ regularly increases (it increases by 1 on each simulation step, and remains unchanged on other steps) and c is not periodic. Hence the result.

We now show that the problem is undecidable even in the case of 2-counter machines. Let M' be a counter machine on n registers $R_1, R_2, \ldots, R_n$. We construct a machine M'' on two registers S and T such that M'' has a periodic configuration if and only if M' has. The values of the registers R_i of M' are stored in the register S of M'' by the classical prime number encoding. The non-negative integers $r_1, r_2, \ldots, r_n$ are encoded into the non-negative integer s by $s = 2^{r_1}3^{r_2}5^{r_3}\ldots\pi(n)^{r_n}$, where $\pi(n)$ is the nth prime number. Incrementation (respectively, decrementation) of the register R_i can then be simulated by multiplying (respectively, dividing) s by $\pi(i)$, and testing whether $R_i = 0$ can be done by testing the divisibility of s by $\pi(i)$. These operations can be performed with just one additional register T. For initial configurations of M'' with S not being an exact product of the required prime powers, the trailing factors remain untouched during all iterations, and so our construction does not introduce any unwanted periodic configuration for M''. On the other hand, the machine M'' clearly has periodic configurations when M' has.

Finally, we prove the decidability for one-counter machines.

Suppose, without loss of generality, that there are no transitions which do not modify the register value (any cycle only having these transitions is trivially decidable, and any other cycle still remains a cycle when taking them off).

We shall prove that a one-counter machine M with n states has a cycle if and only if it has one for which the value of the register is bounded by n. This then trivially ensures the decidability: simply check all $n(n+1)$ possible configurations for periodicity (by iterating M on each of them until either M cycles or until it gets the register value greater than n).

Let us first notice that if $((q_1,r_1),(q_2,r_2),\ldots,(q_{p-1},r_{p-1}),(q_p,r_p))$ is a sequence of configurations that is part of a cycle and such that $r_i>0$ for all i, then:

(i) if $q_p=q_1$, then $r_p\leqslant r_1$;

(ii) if there are n' different states in $\{q_1,\ldots,q_p\}$, then $r_i-r_1<n'$ for all i.

Property (i) is obvious, since otherwise the machine M would increase the register forever. Property (ii) can easily be proven by induction on p. It clearly holds if $p=1$; assume that the property holds for sequences of length less than p, and consider the sequence $((q_1,r_1),\ldots,(q_p,r_p))$. Note that $r_2\leqslant r_1+1$. If q_1 does not occur again in $(q_2,\ldots,q_p)$, then the property can be applied to $((q_2,r_2),\ldots,(q_p,r_p))$, which contains at most $n'-1$ different states, so that $r_i-r_2<n'-1$ for all $i\geqslant 2$, hence $r_i-r_1<n'$. If q_1 occurs again, then let q_j be its second occurrence. By (i), $r_j\leqslant r_1$, and the induction hypothesis can be applied both to $((q_2,r_2),\ldots,(q_{j-1},r_{j-1}))$ (empty if $j=2$) and to $((q_j,r_j),\ldots,(q_p,r_p))$.

Now the main proof follows easily.

Suppose that M has a cycle of length p and consider the infinite and periodic sequence of its configurations

$$((q_1,r_1),(q_2,r_2),\ldots,(q_{p-1},r_{p-1}),(q_p,r_p),(q_{p+1},r_{p+1}),\ldots,(q_{2p},r_{2p}),\ldots).$$

That is, $q_i=q_{p+i}$ and $r_i=r_{p+i}$ for all integers i. Suppose now that there are configurations with zero values for the register. We can then conclude by applying property (ii) for each (finite) subsequence given by a couple (t,z) of integers such that $r_t=r_z=0$ with $t<z$ and for all i with $t<i<z$, $r_i>0$.

Suppose that now the cycle has no zero-valued-register configuration. Then, by finding the index j giving the minimum $r_j=\min_{1\leqslant i\leqslant p} r_i$, we easily see that the sequence $(q_1,r_1-r_j+1),(q_2,r_2-r_j+1),\ldots,(q_{p-1},r_{p-1}-r_j+1),(q_p,r_p-r_j+1)$ is again a cycle respecting the conclusion, by applying property (ii) to its sequence from (q_j,r_j-r_j+1) a round the cycle and back to it. □

3. A counterexample to Kůrka's conjecture

We now describe our counterexample to Kůrka's conjecture. As a starting point, consider the Turing machine K_0 represented in Fig. 2.

The machine has three states and operates on the two-letter alphabet $\{0,1\}$. When in state 1, the machine searches for a 1 to the right of the head. Once it has found one, it changes it into a 0, writes a 1 to the right of it, and moves to state 3 from which it starts a left search for a 1. Once it has successfully completed its search, it

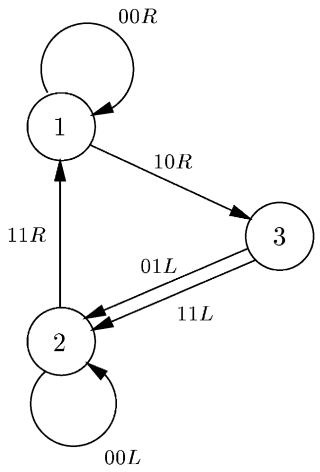

Fig. 2. The Turing machine K_0. The machine is defined by a label-oriented graph. The vertices represent the states of the machine, and the labels on the arcs are used to define the transition function. For example, the label $10R$ on the arc from state 1 to state 3 means that, when in state 1 and reading the symbol 1, the machine writes the symbol 0, shifts the tape one unit to the left (the head thus moves to the right relative to the tape), and then goes into state 3.

returns to state 1. It is easy to verify that this machine does not have any periodic configuration, except for the configurations for which the machine is in its left or right search state (state 1 or 3) and the tape is a tape of 0's. Machines that have search states in fact always have periodic configurations whose associated tape is a periodic tape of symbols that do not satisfy the search. It is therefore clear that in constructing a counterexample we need to eliminate all search states. The machine we construct below essentially performs the same operations as those of K_0, except that it does so by bounding its searches. The technique we use for replacing searches by bounded searches is adapted from [3].

The idea is to simply limit (to an arbitrary, finite constant c) the number of squares the head goes searching for a 1 in vain. Whenever such a search fails, that is the machine discovered c contiguous zeros, then it records the search (as a counter machine when using the stack) and recursively calls a new computation. We then prove that the machine enters a non-periodic sequence of configurations, no matter what the initial configuration it started with.

We give below an implementation of these ideas in C code and an explicit description of a corresponding 36-state loopless Turing machine.

```
int tape[INFINITY];
int i; /* the tape pointer */

void move_right() { /* move a one as far as possible */
  do {
    find_right(); /* find a one to the right, to move it */
```

```
    tape[i]=0;i=i+1;
    if (tape[i]==0) { /* if it is possible to move it */
      tape[i]=1;
      find_left(); /* find a one, to bounce on it */
    }
    else { /* it cannot be moved, so give up */
      find_left(); /* find a one, to clean it */
      tape[i]=0; return;
    }
  } while(1); /* the loop may never end */
}

void find_right() { /* find a one */
  do {
    i=i+1;if(tape[i]) return; /* found it */
    i=i+1;if(tape[i]) return; /* found it */
    i=i+1;if(tape[i]) return; /* found it */
    tape[i]=1;i=i-2;tape[i]=1; /* not found it */
    move_right(1); /* key recursive call,
                      yet which may never return */
  } while(1); /* the loop may never end */
}

void move_left() { /* move a one as far as possible */
  do {
    find_left(); /* find a one to the right, to move it */
    tape[i]=0;i=i-1;
    if (tape[i]==0) { /* if it is possible to move it */
      tape[i]=1;
      find_right(); /* find a one, to bounce on it */
    }
    else { /* it cannot be moved, so give up */
      find_right(); /* find a one, to clean it */
      tape[i]=0; return;
    }
  } while(1); /* the loop may never end */
}

void find_left() { /* find a nonzero to the left */
  do {
    i=i-1;if(tape[i]) return; /* found it */
    i=i-1;if(tape[i]) return; /* found it */
    i=i-1;if(tape[i]) return; /* found it */
        tape[i]=1;i=i+2;tape[i]=1; /* not found it */
            move_left(1); /* key recursive call,
```

```
                    yet which may never return */
  } while(1); /* the loop may never end */
}
```

By looking at the C code above, we notice symmetries and classes of operations: `move_right` calls `find_right` and `find_left`, and symmetrically. In a programming language there is a stack onto which the recursive calls are stored, so that the computation can correctly unfold at return time. For the Turing machine this has to be simulated in the same space, so that the distances do not become too big, and unperiodicity is ensured. A first translation of the program gives a machine we describe next, with two more symbols, 2 and 3, which, together with the 1, are stack markers (the 1 playing a double rôle).

This Turing machine K_1 has 36 states and operates on the four-letter alphabet $\{0,1,2,3\}$. The machine is rather involved, and we show in Fig. 3, only half of the machine; the other half is symmetrical, with L and R being interchanged. In order to describe the behavior of the machine, we have grouped the states in six groups denoted by 1, 2, 3, $1'$, $2'$, $3'$. Each group i has a particular functional purpose, a unique entry state, q_{i1}, and has two exit states: the failed search state q_{i5} and the dispatch state q_{i6}.

Group 1 has the same function as the state 1 of K_0 (search right for a non-zero symbol), group 2 has the same function as the state 2 of K_0 (search left for a non-zero symbol), and state q_{16} corresponds to state 3. In order to bound the searches without introducing periodic configurations we introduce a third group of state (group 3) that has no counterpart in K_0 and construct a symmetric set of states.

The technique for bounding the searches consists in using the searched zone on the tape as a stack. The stack is used to push the group index of failed searches. The corresponding pop operation is performed by groups 3 (for right searches) and $3'$ (for left searches), that have no equivalent among the states of K_0. The pop is a search operation as well, thus a push mechanism for it is also set in place. The state q_{ij} with $j=1,2,3$ perform the bounded search. The state q_{ij} with $j=4,5$ write 1 to be moved by the new search called from q_{i5}, and push the index of the failed search. The return from the new search to the calling search, when the one-unit move fails, is made from q_{16} (respectively, $q_{1'6}$) by moving to q_{31} (respectively, $q_{3'1}$).

In the sequel, configurations of the machine will be given by expressions of the form $({}^{\omega}001\underline{0}100000^{\omega}, q_{11})$. This configuration, for example, is the one of tape content ${}^{\omega}0010100000^{\omega}$, internal state q_{11}, and with the head scanning the underlined symbol.[1] As an illustration, the sequence of configurations obtained when starting from the state q_{11} on a tape of 0's, is given in Table 1.

In this example, we see in the $*$-marked lines that the machine passes through the configurations $({}^{\omega}01\underline{0}0^{n}10^{\omega}, q_{11})$ for $n=0,1$ and 2. In the next section, we prove that

[1] For $n \geqslant 0$, the notation 0^n is used to denote a sequence of n 0s. We use the notation 0^{ω} to denote an infinite sequence of 0s.

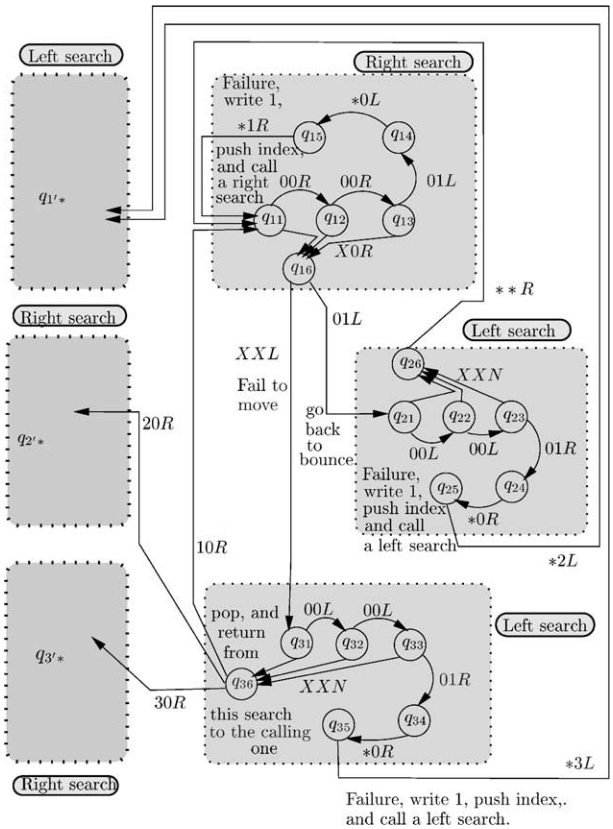

Fig. 3. The Turing machine K_1. Details are given only for half of the machine; the other half is symmetrical. L, R and N mean, respectively, left, right and no tape shift. We use the symbol $*$ to denote an arbitrary symbol of $\{0,1,2,3\}$, and X to denote a symbol from $\{1,2,3\}$.

the machine will then pass through configurations of the type $({}^{\omega}01\underline{0}0^n10^{\omega}, q_{11})$ for increasing values of n, so that the initial configuration $({}^{\omega}0\underline{0}0^{\omega}, q_{11})$ is not periodic; and more generally, whichever configuration it starts from, the machine passes through configurations where the head is at the beginning of increasingly large blocks of zeros, and so the machine may not have periodic configurations.

3.1. K_1 has no periodic configuration

We now prove that the machine K_1 does not have periodic configurations. Due to the symmetry of the machine, the proofs will only be detailed for one half of the configurations, the other being inferred by symmetry. We first need auxiliary definitions and results.

Table 1

Mark	Tape content	Group	State	Purpose
	$^{\omega}00\underline{0}0000000^{\omega}$	1	q_{11}	Search (like state 1 of K_0)
	$^{\omega}000\underline{0}000000^{\omega}$	1	q_{12}	Search
	$^{\omega}0000\underline{0}00000^{\omega}$	1	q_{13}	Search and fail
	$^{\omega}0000\underline{1}00000^{\omega}$	1	q_{14}	Prepare the new search
*	$^{\omega}00\underline{0}0100000^{\omega}$	1	q_{15}	Push the index
	$^{\omega}001\underline{0}100000^{\omega}$	1	q_{11}	Call a right search
	$^{\omega}0010\underline{1}00000^{\omega}$	1	q_{12}	Search and find
	$^{\omega}00100\underline{0}0000^{\omega}$	1	q_{16}	Erase and move (like state 3 of K_0)
	$^{\omega}0010\underline{0}10000^{\omega}$	2	q_{21}	Go back in order to bounce
	$^{\omega}001\underline{0}010000^{\omega}$	2	q_{22}	Search (like state 2 of K_0)
	$^{\omega}00\underline{1}0010000^{\omega}$	2	q_{23}	Search and find
*	$^{\omega}00\underline{1}0010000^{\omega}$	2	q_{26}	Bounce and jump to right search
	$^{\omega}001\underline{0}010000^{\omega}$	1	q_{11}	Search right again (like state 1 of K_0)
	$^{\omega}0010\underline{0}10000^{\omega}$	1	q_{12}	Search
	$^{\omega}00100\underline{1}0000^{\omega}$	1	q_{13}	Search and find
	$^{\omega}0010000\underline{0}000^{\omega}$	1	q_{16}	Erase and move
	$^{\omega}0010\underline{0}01000^{\omega}$	2	q_{21}	Go back
	$^{\omega}001\underline{0}001000^{\omega}$	2	q_{22}	Search
	$^{\omega}00\underline{1}0001000^{\omega}$	2	q_{23}	Search and fail
	$^{\omega}0011\underline{0}01000^{\omega}$	2	q_{24}	Prepare a new left search
	$^{\omega}00110\underline{0}1000^{\omega}$	2	q_{25}	push the index
	$^{\omega}0011\underline{0}21000^{\omega}$	$1'$	$q_{1'1}$	Call a left search
	$^{\omega}0011\underline{0}21000^{\omega}$	$1'$	$q_{1'2}$	Search and find
	$^{\omega}00\underline{1}0021000^{\omega}$	$1'$	$q_{1'6}$	Erase, but fail to move
	$^{\omega}001\underline{0}021000^{\omega}$	$3'$	$q_{3'1}$	Thus, pop the index
	$^{\omega}0010\underline{0}21000^{\omega}$	$3'$	$q_{3'2}$	By searching for it
	$^{\omega}00100\underline{2}1000^{\omega}$	$3'$	$q_{3'3}$	Search and find
	$^{\omega}00100\underline{2}1000^{\omega}$	$3'$	$q_{3'6}$	Read and prepare to return
	$^{\omega}0010\underline{0}01000^{\omega}$	2	q_{21}	Pop, and return to left search group 2
	$^{\omega}001\underline{0}001000^{\omega}$	2	q_{22}	Resume the left search
	$^{\omega}00\underline{1}0001000^{\omega}$	2	q_{23}	Search and find
*	$^{\omega}00\underline{1}0001000^{\omega}$	2	q_{26}	Bounce and jump to the right search
	$^{\omega}001\underline{0}001000^{\omega}$	1	q_{11}	Restart the right search

Definition 1. For $s \geqslant 0$, let $Q(s)$ be the proposition:

"The machine goes from any configuration of the form $(\ldots \underline{0}0^s XY \ldots, q_{11})$, with $X \in \{1,2,3\}$ and $Y \in \{0,1,2,3\}$, to configuration $(\ldots 0^{s+1}0Y \ldots, q_{16})$, where the parts of the tape represented by $\ldots$ remain unchanged. For $i = 2$ and $i = 3$, the machine goes from any configuration of the form $(\ldots X0^s\underline{0}\ldots, q_{i1})$ to configuration $(\ldots X0^{s+1}\ldots, q_{i6})$. The symmetric statements also hold."

For $k \geqslant 2$, let $P(k)$ be the proposition:

"For any integers $t, p, n \geqslant 0$ such that $t + p + n + 2 = k$ and for any $X, Y, Z \in \{1,2,3\}$ the machine goes from configuration $(\ldots X0^t\underline{0}0^p Z0^n Y \ldots, q_{11})$ to configuration $(\ldots X0^k Y \ldots, q_{36})$. The symmetric statement also holds."

Proposition 2. *Let $s \geqslant 0$. If $P(k)$ is true for all integers k with $2 \leqslant k \leqslant s$, then $Q(s)$ is true.*

Proof. The proof is by induction on s. The cases $s = 0$ and $s = 1$ can be checked by hand by following the machine's diagram in Fig. 3.

Then, to prove the proposition by induction on s it is sufficient to show that ($Q(s-1)$ and $P(s)$) imply $Q(s)$. We have the following configuration sequence:

$$
\begin{array}{ll}
\ldots \underline{0}000^{s-2}XY\ldots & q_{11} \\
\quad\downarrow & \text{A few iterations} \\
\ldots 1\underbrace{\underline{0}10^{s-2}_{s}}XY\ldots & q_{11} \\
\quad\downarrow & P(s)\ \text{(hypothesis)} \\
\ldots \underline{1}000^{s-2}XY\ldots & q_{36} \\
\quad\downarrow & \text{One iteration} \\
\ldots 0\underline{0}00^{s-2}XY\ldots & q_{11} \\
\quad\downarrow & Q(s-1)\ \text{(hypothesis)} \\
\ldots 0000^{s-2}0\underline{Y}\ldots & q_{36}
\end{array}
$$

The proof for the configurations $(\ldots \underline{0}0^{s}X\ldots, q_{21})$ and $(\ldots \underline{0}0^{s}X\ldots, q_{31})$ is similar. □

Lemma 1. *$P(k)$ is true for all $k \geqslant 2$.*

Proof. Define $\hat{P}(k,n)$ to be the proposition $P(k)$ where n is also fixed. Let $L = \{(k,n) | n \geqslant 0, n+2 \leqslant k\}$ and consider the lexicographical order $<_L$, i.e., $(k',n') <_L (k,n)$ when $k' < k$, or when $k' = k$ and $n' < n$. The order $<_L$ is well-founded, therefore allowing us to prove the lemma by induction.

Fix (k,n) from L and assume that $\hat{P}(k',n')$ is true for all $(k',n') \in L$ such that $(k',n') <_L (k,n)$. Note that this implies, in particular, that $P(k')$ is true for all $k' < k$ and so, by Proposition 2, $Q(s)$ is true for all $s \leqslant k-1$.

Case $n = 0$. We have the following configuration sequence:

$$
\begin{array}{ll}
\cdots X0^{k-p-2}\underline{0}0^{p}ZY\cdots & q_{11} \\
\quad\downarrow & Q(p)(p \leqslant k-1) \\
\cdots X0^{k-p-2}00^{p}0\underline{Y}\cdots & q_{16} \\
\quad\downarrow & \text{one iteration} \\
\cdots X0^{k-p-2}00^{p}\underline{0}Y\cdots & q_{31} \\
\quad\downarrow & Q(k-1) \\
\cdots \underline{X}0^{k-p-2}00^{p}0Y\cdots & q_{36}
\end{array}
$$

Case $n>0$. We have the following configuration sequence:

$$\begin{array}{ll}
\cdots X0^{k-n-p-2}\underline{00}^{p}Z00^{n-1}Y\cdots\ q_{11} & \\
\qquad\qquad\downarrow & Q(p)\\
\cdots X0^{k-n-p-2}00^{p}0\underline{00}^{n-1}Y\cdots\ q_{16} & \\
\qquad\qquad\downarrow & \text{one iteration}\\
\cdots X0^{k-n-p-2}00^{p}\underline{0}10^{n-1}Y\cdots\ q_{21} & \\
\qquad\qquad\downarrow & Q(k-1-n)\\
\cdots \underline{X}0^{k-n-p-2}00^{p}010^{n-1}Y\cdots\ q_{26} & \\
\qquad\qquad\downarrow & \text{one iteration}\\
\cdots X\underline{00}^{k-n-p-2}0^{p}010^{n-1}Y\cdots\ q_{11} & \\
\qquad\qquad\downarrow & \hat{P}(k,n-1)(\text{induction hypothesis})\\
\cdots \underline{X}00^{k-n-p-2}0^{p}000^{n-1}Y\cdots\ q_{36} &
\end{array}$$

In both cases $\hat{P}(k,n)$ holds. This concludes the proof. □

Putting together Proposition 2 and Lemma 1 we immediately conclude:

Lemma 2. *$Q(s)$ is true for all $s\geqslant 0$.*

We now prove that the machine does not have periodic configurations of a particular type. [2]

Lemma 3. *The configurations $(\cdots\underline{0}0^{\omega},q_{ij})$ with $i\in\{1,2',3'\}$ and $j\in\{1,2,3\}$ are not periodic. This is also true for the symmetric case.*

Proof. Starting from the configuration $(\cdots\underline{0}0^{\omega},q_{ij})$ for some $i\in\{1,2',3'\}$ and $j\in\{1,2,3\}$, the machine goes to configuration $(\cdots X\underline{0}10^{\omega},q_{11})$ where, depending on i, $X=1,2$ or 3. This configuration is of the type $(\cdots X\underline{0}0^{p}Y0^{\omega},q_{11})$ for some $X,Y\in\{1,2,3\}$ and we claim that these configurations are not periodic. Indeed, for any $p\geqslant 0$ we have:

$$\begin{array}{ll}
\cdots X\underline{0}0^{p}Y00^{\omega}\ q_{11} & \\
\quad\downarrow & Q(p)\\
\cdots X00^{p}0\underline{00}^{\omega}\ q_{16} & \\
\quad\downarrow & \text{one iteration}\\
\cdots X00^{p}\underline{0}10^{\omega}\ q_{21} & \\
\quad\downarrow & Q(p)\\
\cdots \underline{X}00^{p}010^{\omega}\ q_{26} & \\
\quad\downarrow & \text{one iteration}\\
\cdots X\underline{0}0^{p}010^{\omega}\ q_{11} &
\end{array}$$

[2] A simulation of this machine can be run from the web page http://www.univ-st-etienne.fr/eurise/nichitiu, link "Software".

We can iteratively apply this result, and, from the obtained configuration sequence, extract a subsequence of configurations of the type $(\cdots X\underline{0}0^p10^\omega, q_{11})$ with strictly increasing values of p. Therefore, the sequence is not periodic. □

In order to prove our main theorem, we need one more lemma.

Lemma 4. *For any $n \geqslant 1$, let E_n be the set of configurations defined by*

$$E_n = \{(\cdots \underline{0}0^{n-1}\cdots, q_{i1}) | i \in \{1, 2', 3'\}\} \cup \{(\ldots 0^{n-1}\underline{0}\ldots, q_{i'1}) | i' \in \{1', 2, 3\}\}.$$

Starting with a configuration from E_n, the machine eventually approaches a configuration from E_{n+1}. Here, contrary to Definition 1, *the parts of the tape represented by $\cdots$ can be modified in the process.*

Proof. We only consider the cases $(\cdots \underline{0}0^{n-1}\cdots, q_{i1})$ for $i = 1, 2', 3'$. Let X be the symbol on the tape to the right of $\underline{0}0^{n-1}$. If $X = 0$, then we are already in a configuration $(\cdots \underline{0}0^n\cdots, q_{j1})$. Let us now assume that $X \in \{1, 2, 3\}$. There are three cases to consider.

Case $i = 1$. We have the following sequence of configurations, where $Z \in \{0, 1, 2, 3\}$:

$\cdots \underline{0}0^{n-1}XZ\cdots\ q_{11}$	line 1A
$\downarrow$	$Q(n-1)$
$\cdots 00^{n-1}0\underline{Z}\cdots\ q_{16}$	
$\downarrow$	one iteration in the case $Z = 0$
$\cdots 00^{n-1}\underline{0}1\cdots\ q_{21}$	
	in the case $Z \in \{1, 2, 3\}$
$\cdots 00^{n-1}\underline{0}Z\cdots\ q_{31}$	

Case $i = 2'$. We have the following sequence of configurations, where $Z, T \in \{0, 1, 2, 3\}$:

$\cdots ZT\underline{0}0^{n-2}0X\cdots\ q_{2'1}$	line 1B
$\downarrow$	$Q(n-1)$
$\cdots ZT00^{n-2}0\underline{X}\cdots\ q_{2'6}$	
$\downarrow$	one iteration
$\cdots ZT00^{n-2}\underline{0}X\cdots\ q_{1'1}$	

if $T \neq 0$, this is exactly symmetrical to the line 1A of the previous table. If $T = 0$, then we are in the desired configuration $(\cdots 0^n\underline{0}\cdots, q_{j'1})$.

Case $i = 3'$. We have the following sequence of configurations, where $Z, T \in \{0,1,2,3\}$:

$$\begin{array}{ll} \cdots ZT\underline{0}0^{n-2}0X\cdots & q_{3'1} \\ \quad\downarrow & Q(n-1) \\ \cdots ZT00^{n-2}0\underline{X}\cdots & q_{3'6} \\ \quad\downarrow & \text{one iteration} \\ \cdots ZT00^{n-2}\underline{0}0\cdots & q_{k1} \end{array}$$

If $T=0$ then we are done, so let us assume that $T \in \{1,2,3\}$. The value of k depends on X. If $X=1$ (respectively, $X=2$), then $k=1'$ (respectively, $k=2$) and we conclude with the symmetry of line 1A (respectively, line 1B). If $X=3$, then $k=3$, and the sequence continues

$$\begin{array}{ll} \cdots ZT00^{n-2}\underline{0}0\cdots & q_{31} \\ \quad\downarrow & Q(n-1) \\ \cdots Z\underline{T}00^{n-2}00\cdots & q_{36} \\ \quad\downarrow & \text{one iteration} \\ \cdots Z0\underline{0}0^{n-2}00\cdots & q_{j1} \quad (\text{some } j) \end{array}$$

which concludes the proof of the lemma. □

We can now finally prove the following theorem.

Theorem 3. *The machine* K_1 *has no periodic configuration.*

Proof. Consider an arbitrary configuration c. After at most 5 steps, the machine enters a state q_{i1} for some $i \in \{1,2,3,1',2',3'\}$. A brief case analysis shows that if the scanned symbol is non-zero, after a few iterations, no matter what the symbols are around the scanned one, the machine returns in a state q_{i1} with a zero as scanned symbol. Then we are in the situation as described in Lemma 4, and for all n the machine will reach a configuration of the form $(\cdots\underline{0}0^{n-1}\cdots, q_{i1})$ with $i \in \{1,2',3'\}$, up to symmetry. At some point, either the machine will reach a configuration of the form $(\cdots\underline{0}0^{\omega}, q_{i1})$, up to symmetry, which by Lemma 3 is not periodic; or we can extract an infinite sequence of configurations $(\cdots\underline{0}0^{n-1}X\cdots, q_{i1})$, up to symmetry, with $X \neq 0$ and increasing n, so that the initial configuration is not periodic. □

3.2. Smaller machines

We can analyze the intrinsic features of K_1 that make it free of periodic configurations, and then compress the machine as much as possible while keeping the observed features. Two compressions are possible. First, the number of steps used in the bounded searches can all be reduced. Second, two groups for each half of the machine almost perform the same kind of search and can be merged. By implementing these two modifications, we obtain the six-state machine K_2 depicted in Fig. 4. We decided to present a formal proof for K_1 and not for this apparently simpler machine K_2, because the

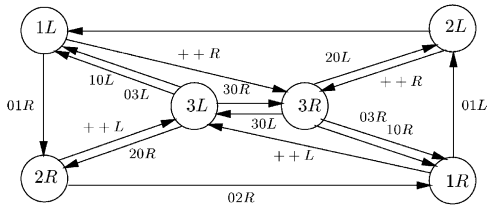

Fig. 4. K_2, a smaller version of K_1. The symbol $+$ represents any letter of the alphabet except 0, thus $++L$ is an abbreviation for $11L, 22L, 33L$.

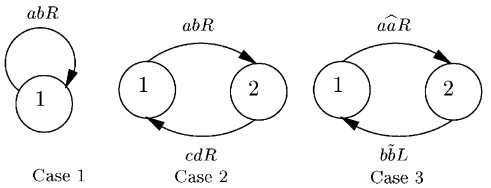

Fig. 5. Three qualitatively distinct cases. In all three cases there exists a periodic configuration.

behavior of the six-state machine is much more complicated and much more difficult to grasp, and so is its proof, while the behavior of K_1 is much closer to the basic initial ideas.

We do not know whether a smaller machine can be found while maintaining the cardinality of the alphabet as 4.

If we give up this provision, then by using the fact that the immortality problem is undecidable for Turing machines with three states [8], it is possible to build a three-state Turing machine with no periodic configuration. This construction is a little involved and we do not present it here; it is essentially the same construction as the one used for defining K_1, but with the three-state machine defined in [8] rather than the machine appearing in [3].

On the other hand,

Theorem 4. *Turing machines with moving tape and no halting states that have only two states always have at least one periodic configuration.*

Proof. For a two-state ($Q=\{1,2\}$) TM on an arbitrary finite alphabet Σ, there are three qualitatively distinct cases, as shown in Fig. 5:

(1) The machine has a loop onto the same state, that is there exist $q\in Q$, $a,b\in\Sigma$ and $D\in\{L,R\}$ with $\delta(q,a)=(q,b,D)$. In this case, the configurations $({}^{\omega}b\underline{a}a^{\omega},q)$, if $D=R$, and $({}^{\omega}a\underline{a}b^{\omega},q)$ if $D=L$, are both periodic configurations of period 1.

Assume now that the machine has no such loop. Yet the machine does not have halting configurations, and so it must have transitions for all (state, symbol) combinations. Thus, the transitions are all transitions that leave one state for the other state. We distinguish two cases.

(2) The machine has a transition from a state to the other and then back, with *the same tape movement*, that is there exist $a,b,c,d \in \Sigma$ with $\delta(1,a)=(2,b,R)$ and $\delta(2,c)=(1,d,R)$, or the same situation, with L instead of R. In this case, the configurations $({}^{\omega}(bd)\underline{a}c(ac)^{\omega},1)$, and, when L exists instead of R, $({}^{\omega}(ca)c\underline{a}(db)^{\omega},1)$, are both periodic configurations of period 2.

(3) The transitions from state 1 to state 2 are all labeled by the same direction symbol (L or R) and the opposite direction symbol appears on all transitions from state 2 to state 1. Without loss of generality assume that R is the direction label on the transitions from state 1 to state 2. Let $g_1 : \Sigma \to \Sigma$ be such that $\delta(1,a)=(2,g_1(a),R)$ and let g_2 be similarly defined. The function g_1 is a permutation of a finite set; therefore, there exist some iteration $g_1^{n_1}$ for it and an element $a \in \Sigma$ for which $g_1^{n_1}(a)=a$. We similarly have $g_2^{n_2}(b)=b$ for some $n_2 \geqslant 1$ and some $b \in \Sigma$. It is then easy to verify that the configuration $(\ldots \underline{a}b \ldots, 1)$ is periodic with a period equal to twice the least common multiple of n_1 and n_2.

We thus conclude the proof. □

This fact does not contradict the presence of universal two-state machines, shown by Shannon (see [9]), since it only claims that at least one among all possible configurations is periodic, thus leaving room for other configurations to start the universal computations. Finally, it is also possible to construct a Turing machine operating on a binary alphabet with no periodic configuration, at the expense of a larger number of states, by encoding elements of $\{0,1,2,3\}$ into binary words of length two.

4. Discussion

Various models of computing devices (Turing machines, counter machines, cellular automata, recurrent artificial neural networks, etc.) are equivalent in terms of computational power when they are seen as defining functions from input to output. In this paper, we consider the *dynamics* of some computing devices. We do not just look at the relations between inputs and outputs, but also look at what happens in between. We also look at configurations that do not correspond to a valid input (e.g., infinite tape content). The answer to the question we consider in this context (the existence of periodic configurations) highly depends on the chosen computing model. For example, recurrent neural networks always have a periodic configuration (a periodic configuration is given by the fixed point for which all activation levels are set equal to 0). For these networks one can also infer from Theorem 1 of [1] the existence of networks that have no other periodic configuration. The situation for cellular automata is quite different. For a given cellular automata, there are only finitely many space-periodic configurations of a given period. Space-periodicity is preserved through iteration, and so all space-periodic configurations of cellular automata are periodic and cellular automata always have infinitely many periodic configurations. In the case of Turing machine we have shown that, contrary to what was conjectured, not all Turing machines have a periodic configuration. Finally, in the case of counter machines we have proved that identifying the presence of a periodic configuration is an undecidable task. These few examples show how rich and different the dynamics of these computing devices are.

Acknowledgements

This research was supported by NATO under grant CRG-961115, by the European Community Fourth Framework Program through the research network ALAPEDES, by the Belgian program on interuniversity attraction poles IUAP P4-02, and by the French Ministry of Education and Research. We are grateful to Pr. Maurice Margenstern and Pr. Petr Kůrka for their careful review and suggestions for improvement.

References

[1] V.D. Blondel, O. Bournez, P. Koiran, J.N. Tsitsiklis, The stability of saturated linear dynamical systems is undecidable, J. Comput. System Sci. 62 (2001) 442–462.
[2] V.D. Blondel, O. Bournez, P. Koiran, Ch.H. Papadimitriou, J.N. Tsitsiklis, Deciding stability and mortality of piecewise affine dynamical systems, Theoret. Comput. Sci. 255 (1–2) (2001) 687–696.
[3] P.K. Hooper, The undecidability of the turing machine immortality problem, J. Symbolic Logic 31 (2) (1966) 219–234.
[4] J.E. Hopcroft, J.D. Ullman, Formal Languages and Their Relation to Automata, Addison-Wesley, 1969.
[5] P. Kůrka, On topological dynamics of turing machines, Theoret. Comput. Sci. 174 (1997) 203–216.
[6] C. Moore, Generalized shifts: unpredictability and undecidability in dynamical systems, Nonlinearity 4 (1991) 199–230.
[7] C. Moore, Finite-dimensional analog computers: flows, maps, and recurrent neural networks, in Proc. First Internat. Conf. Unconventional Models Computation, Auckland, New Zealand, 1998.
[8] Y. Rogozhin, Unsolvability of the immortality problem for Turing machines with three states, Cybernetics (Kibernetika), Kiev (USSR) 1 (1975) 41–43.
[9] C.E. Shannon, A universal Turing machine with two internal states, Automata Studies, Annals of Mathematics Studies, Vol. 34, Princeton University Press, Princeton, NJ, 1956, pp. 157–165.