UDK 519.7

©2008. **O. Kurganskyy, I. Potapov**

Walking Automata on a Class of Geometric Environments

In this paper we study the model of a finite state automaton interacting with two-dimensional geometric environment. We consider a kind of well-known 4-way finite state automaton introduced by Blum and Hewitt and the geometric environments defined as a subspace of two dimensional integer grid bounded by integer value functions. As a main result of this paper we characterize a wide class of geometric environments where a finite automaton exhibits universal behaviour.

1. Introduction. Finite state automata arose as models of transducers of discrete information, i.e. models interacting with their environments [10]. Automata on picture languages [4, 21], automata in labyrinths [3, 11, 12], communicating automata [15], multicounter automata [19, 8], a model of a computer in the form of interaction of a control automaton [5] are examples of such an interaction.

The fundamental problem for systems where an automaton interacts with infinite environment is the reachability problem: "Does a global state S (state of the automaton and state of the environment) belong to the set of states reachable from an initial global state". The reachability problem has connections to many classical problems in automata theory such as diagnostic problems, distinguishability problems, searching in labyrinths, etc. One of the standard methods to show the undecidablity of the reachability problem for some model is to prove that this computational model is universal.

In this paper we consider the computational power of the reactive system, where a finite automaton interacts with a two-dimensional geometric environment.

It was shown in [13, 14] that the problem of checking indistinguishable states for two finite automata in two-dimensional environment E (geometric environment) is decidable if and only if the reachability problem for a finite state automaton in E is decidable. The geometric environment is called *effective* if the reachability problem is decidable and *non-effective* otherwise. It is known from [6, 7, 14] that the environment defined by a set of rectangles of fixed height and the environment represented by a regular or context-free expression are efficient. There are many other classes of effective environments with or without holes that can be constructed by substitution of one cell in effective environment by another effective environment [14].

It was proven in [3] that the sets of input-output words generated by automata interacting with geometric environments without holes, namely, rectangles of unlimited height are, in general, context-sensitive languages. Thus the reachability problem for finite automata interacting with geometric environments is fundamentally difficult and algorithmically unsolvable, in the general case. In spite of known undecidability results it is interesting to identify new classes of non-effective environments, i.e. where a model of finite automaton interacting with these environments is universal.

In this connection we consider a very natural geometric interpretation of the universal model of two counter automata with zero testing (Minsky machine). The Minsky machine has Turing-machine power and can be simulated by a sequence of automaton's (or robot's) movements on the positive quadrant of two dimensional integer grid. In this case the values a and b of two counters correspond to the point (a, b) in the quadrant and the updates of counters correspond to the moves of such automaton to the left, right, up or down. The only testing ability of the automaton is to check whether it is close to the wall or it is not. In such interpretation it is directly corresponds to the zero testing in Minsky machine [20]. We belive that the geometric interpretation could give us much more flexibility about natural restrictions for counter automata and could help with understanding the abilities and limitations of other computational models.

We start our exploration from the environment defined by a quadrant of the plane that corresponds to the Minsky machine model. First we consider an extension of quadrant of the plane, changing the vertical border by power, polynomial or linear functions. In particular we show that the finite automaton interacting with these environments can simulate a Minsky machine. Then we consider an environment which impedes the direct simulation of multiplication. However we show on example of an environment defined by parabola:

$$D_{n^2,n^2} = \{(x, y) \in Z \times Z | y \geq x^2\}$$

that FSA interacting with D_{n^2,n^2} is again a universal model of computation.

Then we generalize the class of non-effective geometric environments. Our conjecture is that the environment in the half plane is non-effective if and only if it is defined by an unbounded nondecreasing function.

The long-term goal of this work is to characterize the whole class of geometric environments to obtain a better understanding of the border between decidability and undecidability for the reachability problems. While such an ambitious goal is not feasible at the moment, we instead investigate several special cases of geometric environments, that we believe is a step towards our ultimate goal.

2. Automata interacting with environments. In what follows we use traditional denotations N, Z and Z_n for the sets of naturals, set of integers and set of bounded integers $\{-n, \ldots, -2, -1, 0, 1, 2, \ldots, n\}$ respectively.

Let $A = (S_A, I, O, \delta_A, \lambda_A, s_0)$ be a finite deterministic everywhere defined Mealy automaton, where S_A, I and O are the sets of states, input symbols, and output symbols, respectively, and $\delta_A : S \times I \to S$ and $\lambda_A : S \times I \to O$ are transition function and function of outputs respectively and $s_0 \in S$ is an initial state.

The geometric environment is defined by possibly infinite (countable) Moore automaton $E = (D, O, I, \delta_E, \lambda_E)$, where $D \subseteq Z \times Z$ is a set of states, $O = Z_1 \times Z_1$ is a set of input symbols, $I = 2^{Z_1 \times Z_1}$ is a set of output symbols, $\delta_E : D \times O \to D$ such that

$$\delta_E((x, y), (d_1, d_2)) = \begin{cases} (x + d_1, y + d_2), & (x + d_1, y + d_2) \in D \\ undefined, & (x + d_1, y + d_2) \notin D \end{cases}$$

is the partial transition function and $\lambda_E : D \to I$ such that

$$\lambda_E(x, y) = \{(i, j) \in Z_1 \times Z_1 | (x + i, y + j) \in D\}$$

is the function of outputs. We represent $\lambda_E(x,y)$ as the matrix $(o_{ij})_{i,j\in Z_1}$, where $o_{ij}=1$ if $(x+i,y+j)\in D$, otherwise $o_{ij}=0$.

We call the set of states D – the nodes of the environment, the symbols of the output alphabet I – the labels of nodes, the function of outputs λ_E – the function of labels of nodes and the words in alphabet O – the movements of automaton in the environment. From Definition 2 follows that two-dimensional geometric environment is represented by an automaton E. According to the fact that the set D uniquely defines the environment E, we identify D and E in the rest of the paper.

Two nodes of the environment (x_1,y_1) and (x_2,y_2) are the neighbours iff $(x_2-x_1,y_2-y_1)\in Z_1\times Z_1$.

Let us describe the process of interaction between automaton and environment. An automaton A initiates the interaction with an environment E starting from the state s_0 and a node $r\in D$. Let A be in state s and node r, then automaton moves to the state $\delta_A(s,\lambda_E(r))$ and the node

$$\delta_E(r,\lambda_A(s,\lambda_E(r))).$$

By pair (s,r), where s is state of an automaton, and r is a node of the environment, we denote a *configuration* of the automaton in the environment. We also say that configuration (s',r') is directly reachable from (s,r) if $s'=\delta_A(s,\lambda_E(r))$ and $r'=\delta_E(r,\lambda_A(s,\lambda_E(r)))$ and denote it by $(s,r)\to(s',r')$.

A geometric environment is called *effective* if the reachability problem for this environment is decidable and *non-effective* otherwise.

3. From Minsky machine to more exotic models of computation. In this section we start from geometric interpretation of a well-known model of two-counter Minsky machine that can increment and decrement counters by one and test them for zero. It is known that Minsky machine is equivalent to Turing machine [20].

It is easy to see that the behaviour of 2-counter machine with a zero test (Minsky machine) can be interpreted as a finite state automaton that interacts with the quadrant of the plane $D_Q=\{(x,y)\in Z\times Z|x\geq 0,y\geq 0\}$. The node (x,y) of the environment $E_Q=(D_Q,O,I,\delta_E,\lambda_E)$ represent the values x and y of two counters and the empty counters of Minsky machine corresponds to the situation when an automaton is on the borders of the geometric environment E_Q (i.e. in a cell of the environment that does not allow to move at least in one direction).

Now we can define some exotic models of computation by changing the shape of the geometric environment. Let us consider an extension of the quadrant of the plane, changing the vertical border by a power function, a polynomial function, a linear function or a sublinear function. Let $a\in N$ then by $D_{a\cdot n}$, D_{n^a} and D_{a^n} we denote the following environments:

$$D_{a\cdot n}=\{(x,y)\in Z\times Z|y\geq a\cdot|x|,x<0;y\geq 0,x\geq 0\}$$

$$D_{n^a}=\{(x,y)\in Z\times Z|y\geq |x|^a,x<0;y\geq 0,x\geq 0\}$$

$$D_{a^n}=\{(x,y)\in Z\times Z|y\geq a^{|x|},x<0;y\geq 0,x\geq 0\}.$$

In particular in the next section we show that a finite state automaton in the following three types of geometric environments $D_{2\cdot n}$, D_{n^2} and D_{2^n} can simulate a Minsky machine.

It is not difficult to see that if the left border of the quadrant is extended by a sublinear function (in other words it cannot be covered by a sector in the half plane), for example:

$$D_{sub} = \{(x, y) \in Z \times Z | y \geq \log |x|, x < 0; y \geq 0, x \geq 0\}$$

then a finite automaton in this environment cannot simulate a universal computational model. This environment actually is not *essentially two-dimensional* environment, i.e. the dynamics in this model can be simulated by a finite state automaton interacting with one-dimensional environment.

Another interesting case is an environment which impedes the direct simulation of multiplication as it can be done in the environment D_Q, for example the environment bounded by a parabola. However we show that a finite state automaton interacting with the environment defined by parabola:

$$D_{n^2,n^2} = \{(x, y) \in Z \times Z | y \geq x^2\}$$

is a universal model of computation.

4. Linear, power and polynomial function extensions. In this section we consider several types of geometric environments with borders defined by integral functions $a \cdot x$, x^a and a^x, $a \in N$, $x \in Z$.

Let us first explain a trick that we use to get an equivalent model of two counter machine where one of the counters is used as a scratchpad. Another, counter holds an integer whose prime factorization is $2^a \cdot 3^b$. The exponents a, b can be thought of as two virtual counters that are being simulated. When the real counter is set to one, it is equivalent to setting all the virtual counters to zero. If the real counter is doubled, that is equivalent to incrementing a, and if it is halved, that is equivalent to decrementing a. By a similar procedure, it can be multiplied or divided by 3, which is equivalent to incrementing or decrementing b. To check if a virtual counter such as a (b) is equal to zero, just divide the real counter by 2 (3), see what the remainder is, then multiply by 2 (3) and add back the remainder. That leaves the real counter unchanged. The remainder will have been nonzero if and only if a (b) was zero.

In the rest of the paper we are going to use the universal model of 2-counter machine M_{scrt} with one scratchpad counter to show several undecdiability results for walking automaton in different geometric environments.

The simple extension of machine M_{scrt} shows that the environment limited by two linear functions in the half plane is non-effective. Since the integral line defines the regular shift of the borders, we only need to amortise this shifts by adding a fixed number of right or left moves according to the chosen direction to each transition of original automata. So we can directly simulate multiplication and division that corresponds to original operation of machine M_{scrt}.

Theorem 1. *The geometric environments D_{2n}, D_{n^2}, D_{2^n} are non-effective.*

Proof. The proof of this fact is based on a simulation of Minsky machine by a finite state automaton in the above geometric environments.

Let the finite state machine A interacting with D_Q can reach the configuration (s, a', b') from configuration (s, a, b) by one step. Now we can construct another FSA A' interacting with D_Q, that can reach the configuration $(s, 2^{a'} \cdot 3^{b'}, 0)$ from configuration $(s, 2^a \cdot 2^b, 0)$ by a finite number of states.

To check if a virtual counter such as a (b) is equal to zero, just divide the real counter by 2 (3), see what the remainder is, then multiply by 2 (3) and add back the remainder. That leaves the real counter unchanged. The remainder will have been nonzero if and only if a (b) was zero.

The geometric environment D_{2n}. The straightforward modification of the FSA A' gives the result for any sector environment formed by two lines in the half plane. Since the integral line defines the regular shifts of the border, we only need to amortise these shifts by adding a fixed number of right or left moves according to the chosen direction.

The geometric environment D_{n^2}. The case where one of the borders does not have a periodic shifts is less trivial. However we use again the same scheme and prove that interaction of finite state automaton A' with the quadrant environment D_Q can be reduced in some sense to the interaction of finite state automaton B with environment D_{n^2}. In case of nonperiodic border we need to choose another method to code the counter in the new environment, and more sophisticated method of amortisation during the operations of multiplication and division.

Let the boundary point $(-x, x^2) \in G_{n^2}$ represents a number x. Now let us show that we can convert any point $(-x, x^2)$ to the point $(-2x, (2x)^2)$, that stands for multiplication of x by 2.

First, FSA converts point $(-x, x^2)$ to point $(x^2 - x, 0)$ by moving right and down until it reaches the border. Then it converts the point $(x^2 - x, 0)$ to the $(-2x, 4x^2 + 4x)$ by repeating the following pattern "moving four times up and one time left" until it reaches the border. Since the inequality $(2x)^2 \le 4x^2 + 4x \le (2x + 1)^2$ holds for any natural number we can state that the point $(-2x, 4x^2 + 4x)$ belongs to the border G_{n^2}. Finally the point $(-2x, 4x^2 + 4x)$ can be converted to the point $(-2x, 4x^2)$ by a finite number of moves down along the border, since all points $\{(-2x, w) | (2x)^2 < w \le (2x + 1)^2\}$ are the boundary points. Let us consider how to perform multiplication on 3 in this environment. In other words we need to show that from any point $(-x, x^2)$ we can reach the point $(-3x, (3x)^2)$. We start just as in previous case. We convert point $(-x, x^2)$ into $(x^2 - x, 0)$ and then to $(-3x - 3, 9x^2 + 18x + 27)$, which is the boundary point according to the inequality $(3x + 3)^2 < 9x^2 + 18x + 27 < (3x + 4)^2$, which holds for all $x \in N$. After that FSA moves from point $(-3x - 3, 9x^2 + 18x + 27)$ to point $(-3x, (3x)^2)$ along the border via 3 corners $(-3x - 3, (3x + 3)^2)$, $(-3x - 2, (3x + 2)^2)$ and $(-3x - 1, (3x + 1)^2)$. In a similar way we construct automaton that can divide on 2 and 3 in D_{n^2}.

The geometric environment D_{2^n}. Let us prove that we can perform the operations of multiplication and division by a finite state automaton in the geometric environment D_{2^n}.

Let the boundary point $p = (-\lfloor \log_2 x \rfloor, x)$ stands for a positive number x in the

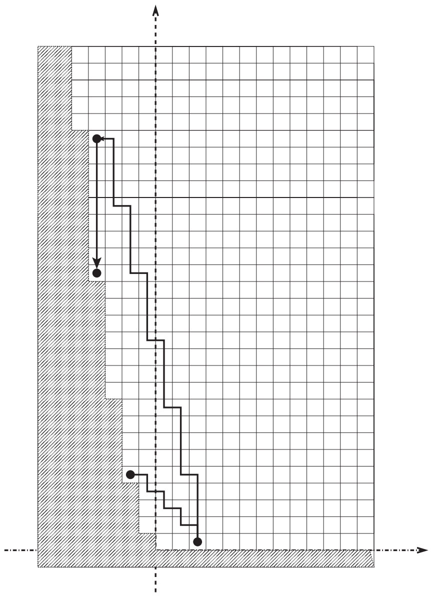

Figure 1. Simulation of the multiplication by 2 in the geometric environment D_{n^2}.

geometric environment D_{2^n}. The finite automaton can move from the point $(-\lfloor \log_2 x \rfloor, x)$ to the new point $(0, x - \lfloor \log_2 x \rfloor)$ by repeating a pair of operations move left and down until it reaches the border. Then the finite automaton can reach the border point r by repeating two moves up and one move left. Thus, r is either $(-\lfloor \log_2 x \rfloor, 2x)$ or $(-\lfloor \log_2 x \rfloor - 1, 2x + 2)$.

Now let us show how to check by a finite automaton in which part of E it reaches the boundary point r from the initial point p. We use the simple property that if n is an even number then $(2n \bmod 4) = 0$ and $(2n + 2 \bmod 4) \neq 0$, if n is an odd number then $(2n \bmod 4) \neq 0$ and $(2n + 2 \bmod 4) = 0$. We first check by finite automaton if the ordinate of the point p is odd or even and then we check the ordinate of the point r on divisibility by 4. So if r's ordinate is divisible on 4 and p's ordinate is odd then the automaton is in the point $(-\lfloor \log_2 x \rfloor - 1, 2x + 2)$ and it can move down to the point $(-\lfloor \log_2 x \rfloor - 1, 2x)$ $=(-\lfloor \log_2 2x \rfloor, 2x)$, otherwise it is in the point $(-\lfloor \log_x \rfloor, 2x)$ $=(-\lfloor \log_2 2x \rfloor, 2x)$.

In a similar way we can show that finite automaton can multiply on 3 and divide on 2 and 3 in the geometric environment D_{2^n}. $\square$

Theorem 2. *The geometric environment* $D_{n^2,n^2} = \{(x, y) \in Z \times Z | y \geq x^2\}$ *is non-effective.*

Proof. Let the boundary point (x, x^2) stands for a positive number x in the geometric environment D_{n^2,n^2}. A finite automaton cannot multiply in D_{n^2,n^2} if it touches the border only by a constant number of times as we have in case of D_{n^2} or D_{2^n}. So in case of D_{n^2,n^2} we introduce some kind of cycle that will be used for multiplication and division.

Let us prove that a finite automaton can reach a point $(2x, (2x)^2)$ from a point (x, x^2).

A finite automaton can reach the boundary point $(-(x+1),(x+1)^2) = (-(x+1), x^2 + 2x+1)$ in D_{n^2,n^2} from a point (x, x^2) by repeating the pattern "up and left", since it will move left for exactly $2x+1$ cells. It is easy to see that an automaton will reach the point $(-(x+1),(x+1)^2+c)$ from (x,x^2+c) for any $0 \le c < 2x+1$ by repeating the same pattern "up and left". Then it can reach the point $(x+1,(x+1)^2+c)$ by moving "right" until it reaches the border and the point $(x+1,(x+1)^2+c+4)$ doing additional 4 moves "up".

Let us call the sequence of moves from the point (x, x^2+c) to the point $(x+1,(x+1)^2+c+4)$ a *cycle*. In the above procedure of multiplication by 2, starting from the point (x, x^2) the finite automaton will meet the cell

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

for the first time exactly after x cycles since it corresponds to the point $(2x, 4x^2+4x)$. If the automaton will move down from the cell

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} \text{ to } \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

it reaches the point $(2x, 4x^2)$ and the multiplication by 2 is completed (see Figure 2). Similarly we can perform multiplication by 3 if we slightly change the previous procedure. If we change our cycle in a such way that the automaton will do 3 additional moves up

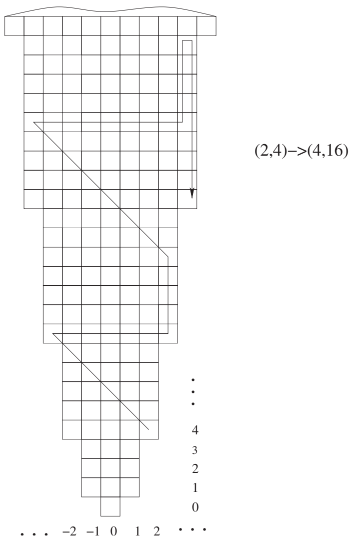

Figure 2. An example of multiplication of $x=2$ by 2 in the geometric environment D_{n^2,n^2}. The operation of multiplication is simulated by the movement of a finite automaton from (x, x^2) to $(2x,(2x)^2)$.

instead of 4 it reaches the cell

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

for the first time after $2x$ cycles and will be in the position $(3x, 9x^2 + 6x)$. After that it moves down in the same way as in previous case to reach the point $(3x, (3x)^2)$. Another pair of operation such that division by 2 and 3 an automaton can perform in the following way. The automaton moves from point (x, x^2) to the point $(-x, x^2)$ and back. In such way it perform $4x$ steps or moves. In order to check the divisibility of x by 2 (or 3) it needs to check the divisibility of $4x$ by 8 (or 12) that can be done using a finite memory. $\square$

In the next section we show a wider class of environments where the reachability for walking automaton is undecidable but the direct multiplication and division are not implementable.

5. Widening Non-effective Environments. Let us consider a function $f : N \to N$ such that $f(n) \geq n$ and function $g : N \to N$ such that $g(1) = 1$, $g(n) = g(n-1) + f(n-1)$. We define now a geometric environment $E(g)$, that is defined by the set of vertices $\{(x, y) \subseteq N \times N | x \geq g(y)\}$. The example of an environment is shown on Figure 3.

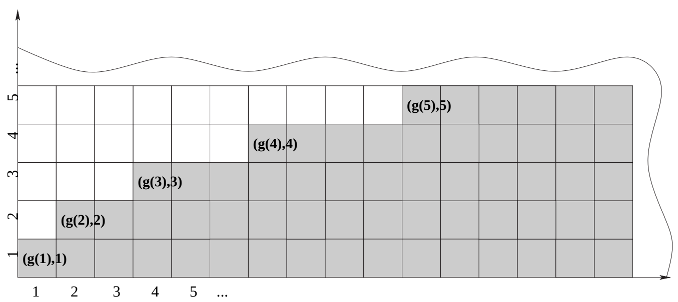

Figure 3. The example of an environment represented by function $f(x) = x$.

Theorem 3. *Given a number $p \in N$. If for all $n > p$ a function f satisfies the condition $f(n) \equiv p \pmod{n}$ then an environment $E(g)$ is non-effective.*

Proof. The proof is done by reduction of the reachability problem for 2-counter automaton (Minsky machine) to the reachability problem for walking finite state automaton in the above environment. In particular we show how to perform operation of multiplication and division in the environment $E(g)$ that allow us to perform the proposed reduction. We show the simulation of multiplication by 2 in the environment $E(g)$ in details. It is easy to apply this method to simulate the multiplication by 3 and division by 2 or 3 by walking automaton in $E(g)$.

Let us assume that an automaton starts from a point $(g(y), y)$ in the environment

$E(g)$. The vertex of environment $E(g)$ in this point has a label

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Now we show that a finite state automaton on environment $E(g)$ can reach a vertex $(g(2y), 2y)$ from a vertex $(g(y), y)$ and stop on it. Note that all movements (computations) of a finite state automaton that interacts with an environment $E(g)$ is done using only labels of environments that are available to our walking automaton and its finite number of states.

By $S_n = (1,0)^n \in O^*$, $n \in N$ we denote the movement of automaton from a vertex (x, y) to a vertex $(x+n, y)$, that is n movements to the right.

Another kind of movements that we are going to use for simulation are movements of type C : C_1 or C_0.

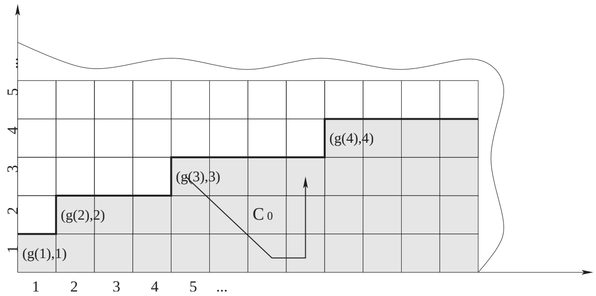

Figure 4. The example of a movement C_0.

If $g(y) \leq x < g(y+1) - y$ by $C_0 = (1,-1)^{y-1}(1,0)(0,1)^{y-1} \in O^*$ we denote the movement of an automaton from a vertex (x, y) to a vertex $(x+y, y)$.

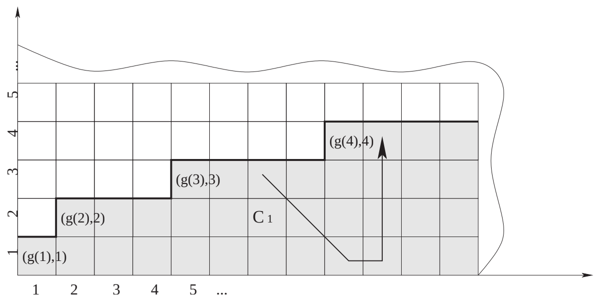

Figure 5. The example of a movement C_1.

If $g(y+1) - y \leq x < g(y+1)$ by $C_1 = (1,-1)^{y-1}(1,0)(0,1)^{y} \in O^*$ we denote the movement of automaton from vertex (x, y) to a vertex $(x+y, y+1)$. The examples of movements C_0 and C_1 are shown on Figures 4, 5.

There are several useful properties of C-movements. Let us assume that a walking automaton is in the bounded vertex (x, y) of an environment $E(g)$, where $g(y) \leq x < g(y+1)$, then it can make only a finite number of consecutive moves of type C_0. If $f(y) = iy + p$ (note that p is a constant), then the number of consecutive moves of type C_0 is bounded by i. So finally after a finite number of C_0 movements a walking automaton does C_1 movement. It can also determine the fact of changing from C_0 to C_1 by checking evenness/oddness of second component of the vertex coordinate using a finite number of its internal states.

Next type of movements that we use for simulation is T. Let a walking automaton is at the boundary vertex (x, y), $g(y) \leq x < g(y) + y$ of an environment $E(g)$, in which the following property $f(y) = iy + p$ holds. By T we denote the sequence of movements from a vertex (x, y) to $(x + p, y)$ by S_p and then a finite number of C_0 movements that end by C_1. Let us prove now that T will move an automaton from a vertex (x, y) to a vertex $(x', y + 1)$, where

$$x - g(y) = x' - g(y + 1) \tag{1}$$

Since we have that $f(y) = iy + p$, a walking automaton should make exactly $i - 1$ movements of type C_0 to reach vertex $(x + (i - 1)y + p, y)$ from vertex $(x + p, y)$. Then after movement C_1 it reaches vertex $(x+(i-1)y+y+p, y+1)$ that is $(x+iy+p, y+1)$. From it follows that $x'-g(y+1) = x+iy+p-g(y+1) = x+iy+p-g(y)-f(y) = x-g(y)$. In other words, we proved that after a movement T the distance from a corner $(g(y), y)$ of the environment to a vertex (x, y) is the same as the distance from a vertex $(x', y+1)$ to a corner $(g(y+1), y+1)$. So we can keep some information of the computations during the walk of automaton as a distance from its location to the nearest left corner of the environment.

Let us show how to move an automaton from a vertex $(g(y), y)$ to a vertex $(g(2y), 2y)$ that corresponds to the multiplication of y by 2. The automaton starts in a vertex $(g(y), y)$ by making a sequence of movements (TS_2) (that will be exactly y such patterns) until the automaton reaches a vertex $(g(2y) + 2y, 2y)$. In fact the automaton can easily recognize whether a point V is of type $(g(x) + x, x)$ by making the moves that are reversible to the movement of type C (i.e an automaton moves down to the wall and then iteratively moves left and up) starting from V. Actually if V is a vertex of type $(g(x) + x, x)$ the automaton should appear in the vertex $(g(x), x)$ with the label

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

In order to come back to the vertex V the automaton can perform one standard movement of type C and continues its predefined actions. So if the current vertex is of the form $(g(2y) + 2y, 2y)$ the automaton makes a number of movements $(-1, 0)$ until it reaches

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

that is exactly the vertex $(g(2y), 2y)$.

Let us track the sequence of above movements. The first movement TS_2 will change the position of automaton from $(g(y), y)$ to $(g(y+1)+2, y+1)$. The position of automaton after next movements TS_2 will be $(g(y+2)+4, y+2)$. Thus, the position of automaton will be changed to $(g(y+y)+2y, y+y) = (g(2y)+2y, 2y)$ exactly after y of such movements. Then it moves to the point $(g(2y), 2y)$ by the sequence of moves $(-1, 0)^{2y}$ until it reaches the corner of the environment. The example of such movements that correspond to the multiplication by 2 is shown on Figure 6.

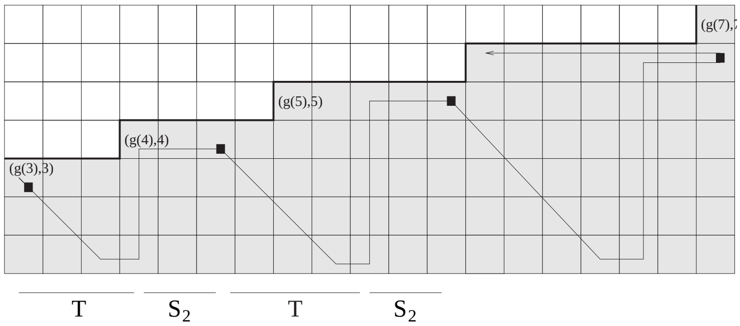

Figure 6. The example of a multiplication $y \times 2$, where $y = 3$, in the environment represented by function $f(x) = x$.

By analogy with a case of multiplication by 2 one can show a similar sequence of movements that that changes the location of automaton from $(g(y), y)$ to $(g(3y), 3y)$ and correspond to multiplication by 3. The sequence of movements that guarantees the division by 2 and 3 can be derived from the operation of multiplications. The question whether a number y is divisible by 2 (3) can be checked by automaton in a standard way using 2 (3) states by moving down from $(g(y), y)$ to the vertex $(g(y), 1)$ and up again. It follows that the walking automaton interacting with an environment $E(g)$ can simulate the computation of two counter machine where one of the counters is used as a scratchpad. Thus the reachability problem of a walking automaton in environment $E(g)$ is undecidable and the model is universal. $\square$

Now we can derive a variety of different non-effective environments from above theorem.

Corollary 1. *Let $g : N \to N$ be an integral polynomial function of degree n, where $n \geq 2$. The environment $E(g)$ is non-effective.*

Proof. Let $g(x) = a_n x^n + \ldots + a_1 x + a_0$ and f(x)= g(x+1)-g(x). Since f is also polynomial, i.e. $f(x) = b_n x^n + \ldots + b_1 x + b_0$, then $f(x) \equiv b_0 \pmod{x}$. It follows form Theorem 3 that the environment $E(g)$ is non-effective. $\square$

Moreover we can change the main theorem, but using the same ideas of computations with respect to some modulo.

Corollary 2. *Given a constant $p \in N$. Let function f fulfil condition $f(n) \equiv n - p \pmod{n}$ for all $n > p$ then an environment $E(g)$ is non-effective.*

Corollary 3. *Given a constant $p \in N$. Let function f fulfil condition $f(n) \equiv \lfloor \frac{n}{2} \rfloor \pmod{n}$ for all $n \in N$ then an environment $E(g)$ is non-effective.*

Corollary 4. *Given a function $f : N \to N$ that can be represented as $f_1(x) \cdot f_2(x)$, where $f_1(n) \equiv 0 \pmod{n}$ and $f_2(n)$ is any nondecreasing function. An environment $E(g)$ is non-effective.*

We can also use Theorem 3 to show that for any environment limited by unbounded nondecreasing function there exist a sub-environment where a finite automaton exhibits universal behaviour:

Corollary 5. *Let $E(h) = \{(x, y) \subseteq N \times N | x \geq h(y)\}$ and $h : N \to N$ is an unbounded integral nondecreasing function. Then there is a function $g : N \to N$, such that $E(g) \subseteq E(h)$ and $E(g)$ is non-effective.*

Proof. Let $g : N \to N$ be a function such that $g(1) = 1$, $g(x) = g(x-1) + f(x-1)$ and let $f(x) = x \cdot c(x)$, where $c(x) : N \to N$ is large enough to satisfy the property $f(x) > h(x+1) - h(x)$. From the above construction follows that $E(g) \subseteq E(h)$. On the other hand the fact that $f(x) \equiv 0 \pmod{x}$ gives us that the environment $E(g)$ is non-effective by Theorem 3. □

6. Conclusion and Discussion. The straightforward consequence of the above result can be stated as follows. We can define a model of two-counter automaton with restricted counters. i.e. the model of two-counter automaton where the maximum value of one counter at every moment is bounded by a some function from the value of another counter. In case if such (even very slow growing) function is of type defined in Theorem 3 we still have a universal model with some slowdown for simulation of every action of classical Minsky Machine.

The decidability of reachability problem for automaton interacting with environments defined by other functions is still an open question. Though we show in Corollary 4 that bounds on increase rate of the environment width does not have any influence on universality, it is not clear what would be the cornerstones of undecidability for some specific environment, e.g. when environment is limited by a logarithmic function: $E_{\log} = \{(x, y) \in N \times N | y \leq \log(x)\}$.

1. *Ben-Amram A.M.* A Complexity-Theoretic Proof of a Recursion-Theoretic Theorem. ACM SIGACT News, Vol.35. – No.2 – 2004 – P.111-112.
2. *Manuel Blum, Carl Hewitt* Automata on a 2-Dimensional Tape. FOCS. – (1967). – P.155-160.
3. *Grunskaya V.I.* Some properties of trajectories of automata in labyrinths, Thesis [in Russian]. – Moscow State University, Moscow. – (1994).
4. *Dora Giammarresi* Finite State Recognizability for Two-Dimensional Languages: A Brief Survey. Developments in Language Theory. – (1995). – P.299-308.
5. *Glushkov V.M., Tseitlin G.E. and Yushchenko E.L.* Algebra, Languages and Programming [in Russian]. – Naukova Dumka, Kiev (1989).
6. *Grunsky I.S., Kurganskyy A.N.* Indistinguishability of finite automata interacting with an environment [in Russian]. – Dokl. AN Ukraine. – v.11 – (1993). – P.31-33.
7. *Grunsky I.S., Kurganskyy A.N.* Indistinguishability of finite automata with constrained behaviour, Cybernetics and systems analysis. – v.5. – (1996). – P.58-72.
8. *Jantzen M. and Kurganskyy O.* Refining the hierarchy of blind multicounter languages and twist-closed trios. – *Information and Computation.* – 185 (2003). – P.159-181.

9. *Katsushi Inoue, Itsuo Takanami* A survey of two-dimensional automata theory. Inf. Sci. – 55(1-3). – (1991). – P.99-121.
10. *Kudryavtsev V.B., Alyoshin S.V. and Podkolzin A.S.* An introduction to the Theory of Automata [in Russian]. – Nauka, Moscow. – (1985).
11. *Kudryavtsev V. B., Ushchumlich Sh. and Kilibarda G.* On the behavior of automata in labyrinths. – Discrete Math. and Applications, 3 (1993). – P.1-28.
12. *Kilibarda G., Kudryavtsev V. B., Ushchumlich Sh.* Collectives of automata in labyrinths Discrete Mathematics and Applications. – 13 (5). – (2003). – P.429-466.
13. *Kurganskyy A.N.* Indistinguishability of the finite automata interacting with an environment. – Thesis [in Russian]. – Saratov State University, Russia. – (1997).
14. *Kurganskij A.N.* Indistinguishability of finite-state automata with respect to some environments. – (Russian, English) [J] 37, No.1. – (2001). – P.33-41; translation from Kibern. Sist. Anal. – 2001. – No.1. – (2001). – P.43-55.
15. *Kurganskyy A.N., Potapov I.G.* On the bound of algorithmic resolvability of correctness problems of automaton interaction through communication channels. – *Cybernetics and System Analysis.* – 3. – (1999). – P.49-57.
16. *Kurganskyy O., Potapov I.* On the computation power of finite automata in two-dimensional environments. – Developments in Language Theory 2004, LNCS 3340. – P.261-271.
17. *Kurganskyy O. , Potapov I.* Universality of Walking Automata on a Class of Geometric Environments. Proceedings of the First Conference on Computability in Europe: New Computational Paradigms, Amsterdam, ILLC Publications X-2005-01. – 2005. – P.122-131.
18. *Markov A.A.* On Normal Algorithms Which Compute Boolean Functions. – Soviet Math. Dokl. 5, 1964, P.922-924.
19. *Maurice Mergenstern* Frontier between decidability and undecidability: a survey. Theoretical Computer Science, v.231. – (2000). – P.217-251.
20. *Minsky M.* Computation: Finite and Infinite Machines. Englewood Cliffs, N.J.: Prentice-Hall. – (1967).
21. *Moore C. and KariJ.* New Results on Alternating and Non-Deterministic Two-Dimensional Finite-State Automata. – International Symposium on Theoretical Aspects of Computer Science, STACS. – (2001). – P.396-406.
22. *Tokio Okazaki, Katsushi Inoue, Akira Ito, Yue Wang* Space Hierarchies of Two-Dimensional Alternating Turing Machines, Pushdown Automata and Counter Automata. – IJPRAI 13(4). – (1999). – P.503-521.

Institute of Applied Mathematics and Mechanics of NASU, Donetsk
University of Liverpool, UK
kurgansk@iamm.ac.donetsk.ua
potapov@liverpool.ac.uk

Received 20.11.08