Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

Annales UMCS Informatica AI 2 (2004) 101-114

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Three simulations of Turing machines with the use of real recursive functions

Monika Piekarz*

Department of Computer Science, Institute of Mathematics, Maria Curie-Skłodowska University, pl. M. Curie-Skłodowskiej 1, 20-031 Lublin, Poland

Abstract

Three simulation algorithms of Turing machines by means of real recursive functions are proposed. Moore's shifting mapping *GS* is used to this end. The relationship between a simulation dimension and classes of η -hierarchy is established.

1. Introduction

The well known models of effective computability such as Turing machines (1936), Post's algorithms (1943), partial recursive functions (1931), the Markow normal algorithms (1954), Church's λ – calculus (1936), the Sheperdson – Sturgis random – access machines (1963) and unlimited register machines were previously introduced. All the models of effective computability are equivalent mutually with respect to their computational abilities.

The Turing machines were the most useful model to point out complexity classes of some problems. Lately several types of Turing machines, which can solve some problems undecidable by the classical Turing machines, were introduced [1-3].

Quite other investigations are related to the real recursive functions. The reason for these studies is first to give a model of analog computation, and second to obtain analog characterization of classical complexity classes. It has been also shown that classical halting problem is analog solvable [4]. This paper deals with a simulation problem of classical Turing machines by means of real recursive functions [5]. The three simulation algorithms by means of recursive functions proposed here are detailed versions of Moore's work [6]. Such simulation is the first step for the better analysis of complexity classes as well as a problem of nondeterminism.

* *E-mail address*: mpiekarz@golem.umcs.lublin.pl

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

2. Basic notions

Let us recall at the beginning the well known notion of Turing machine which will be used in further considerations.

Definition 2.1. The Turing machine TM is defined as the following tuple: $(\Sigma, Q, \delta, q_0, q_f)$; where $\Sigma = \{s_0, s_1, \ldots, s_m\}$, $m \geq 1$, and $Q = \{q_0, q_1, \ldots q_k\}$, $k \geq 1$, are the set of type symbols and the states, $\delta : \Sigma \times Q \to \Sigma \times Q \times \{-1, +1\}$ is a partial function called a transition function, and -1 and +1 are the symbols which indicate the left and right side movement of the head, q_0 and q_f denote the initial and final states, respectively.

We denote a temporary description of the Turing machine with: $\alpha_1 q \alpha_2$, where $q \in Q$ is the current state TM, and $\alpha_1 \alpha_2$ is the two-side infinite string written in the tape. We suppose that the head observes the first from the symbol of the string α_2.

We define the Turing machines work as follows way. Let $\ldots x_1 x_2 \ldots x_{i-1} q x_i \ldots x_n \ldots$ be the Turing machines temporary description. For $\delta(q, x_i) = (p, y, -1)$ we have: $\ldots x_1 x_2 \ldots x_{i-1} q x_i \ldots x_n \ldots \to \ldots x_1 x_2 \ldots p x_{i-1} y \ldots x_n \ldots$; however for $\delta(q, x_i) = (p, y, +1)$ we have: $\ldots x_1 x_2 \ldots x_{i-1} q x_i \ldots x_n \ldots \to$ $\to \ldots x_1 x_2 \ldots x_{i-1} y p \ldots x_n \ldots$.

Now let us recall the basic definition relating to the recursive functions over reals. In particular, we describe η-hierarchy which gives the number of nesting limits in the definition of a given function. This hierarchy is a tool to describe the computational hardness of function. The class of real recursive functions has been introduced by Moore [5], it is a generalization of natural recursive functions [7, 8] to the real numbers and then modified by Mycka and Costa in [4]. Let us recall a definition from [4].

Definition 2.2. The set of real recursive vectors is the least set generated from the real recursive scalars 0, 1, -1 and the real recursive projections $I_i^n(x_1, \ldots, x_n) = x_i$, $1 \leq i \leq n$, $n > 0$, by the operators:

1. Composition: if *f* is a real recursive vector with n k-ary components and *g* is a real recursive vector with k m-ary components, then the vector with n m-ary components $(1 \leq i \leq n)$

$$\lambda x_i \ldots x_m . f_i\left(g_i(x_1, \ldots, x_m), \ldots, g_k(x_1, \ldots, x_m)\right)$$

   is real recursive.
2. Differential recursion: if *f* is a real recursive vector with *n* *k*-ary components and *g* is a real recursive vector with *n* *k*+*n*+1-ary components,

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

then the vector h of n k+1-ary components which is the solution of the Cauchy problem for $1 \le i \le n$

$$h_i(x_1,\ldots,x_k,0) = f_i(x_1,\ldots,x_k),$$
$$\partial_y h_i(x_1,\ldots,x_k,y) = g_i(x_1,\ldots,x_k,y,h_1(x_1,\ldots,x_k,y),\ldots,h_n(x_1,\ldots,x_k,y))$$

is real recursive whenever h is of the class C^1 on the largest interval containing 0 in which a unique solution exists.

3. Infinite limits: if f is a real recursive vector with n k+1-ary components, then the vectors h, h', h'' with n k-ary components $(1 \le i \le n)$

$$h_i(x_1,\ldots,x_k) = \lim_{y\to\infty} f_i(x_1,\ldots,x_k,y),$$
$$h_i'(x_1,\ldots,x_k) = \liminf_{y\to\infty} f_i(x_1,\ldots,x_k,y),$$
$$h_i''(x_1,\ldots,x_k) = \limsup_{y\to\infty} f_i(x_1,\ldots,x_k,y)$$

are real recursive, whenever these limits are defined for all $1 \le i \le n$.

4. Arbitrary real recursive vectors can de defined by assembling scalar real recursive components of the same arity.
5. If f is a real recursive vector, then each of its components is a real recursive scalar.

The first important remark to the above definition is connected with a cardinality of the set of real recursive functions. Because every function has at least one finite syntactical description, hence the number of real recursive functions is countable. In this way we can observe that the system of functions given by our definition is constructive and not too large (not all real functions are captured by it, and, in fact, an uncountable number of real functions is left outside).

Let us discuss carefully the details of the definition. For a differential recursion we restrict a domain to an interval of continuity. This will preserve the analyticity of functions in the process of defining.

The natural measure of a function difficulty can be joined with a degree of discontinuity. The above considerations lead us to the conception of η-hierarchy which describes the level of nesting limits in the definition of a given function.

We should start with the notion of syntactic n-ary descriptions of real recursive vectors. Let us introduce some kind of symbols called basic descriptors for all basic real recursive functions. The combination of such descriptions for given real recursive functions will form a new description of another function. Let us start with the basic functions: i_k^j as a k-ary description for projection I_k^j for all $1 \le i \le k$; $1_k, \overline{1}_k, 0_k$ are k-ary descriptions for constants 1, -1, 0 used with k variables. We must add also operator symbols (descriptors) for all introduced

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

operators: *dr* – for a differential recursion, *c* – for a composition, *l*, *ls*, *li* for a respective kind of limits (lim, lim sup, lim inf).

Now the collection of descriptions of real recursive vectors can be inductively defined as follows: $i_n^j, 1_n, \overline{1}_n, 0_n$ are *n*-ary descriptions of I_n^j, $1 \le j \le n \in N$, $\lambda x_1,\ldots,x_n.1$, $\lambda x_1,\ldots,x_n.-1$, $\lambda x_1,\ldots,x_n.0$ for all $(x_1,\ldots,x_n) \in R^n$, $n \in N$, respectively. If $\langle h \rangle = \langle h_1,\ldots,h_m \rangle$ is a *k*-ary description of the real recursive vector *h* and $\langle g \rangle = \langle g_1,\ldots,g_k \rangle$ is a n-ary description of the real recursive vector *g*, then $c(\langle h \rangle, \langle g \rangle)$ is a *n*-ary description of the composition of *h* and *g*. For differential recursion we can write: if $\langle h \rangle = \langle h_1,\ldots,h_n \rangle$ is a *k*-ary description of the real recursive vector *h* and $\langle g \rangle = \langle g_1,\ldots,g_n \rangle$ is a *k*+*n*+1-ary description of the real recursive vector *g*, then $dr(\langle h \rangle, \langle g \rangle)$ is a *k*+1-ary description of the solution of the Cauchy problem for *h*, *g* (if such a solution exists). Finally, if $\langle h \rangle = \langle h_1,\ldots,h_m \rangle$ is a *n*+1-ary description of the real recursive vector *h*, then $l(\langle h \rangle)$, $li(\langle h \rangle)$, $ls(\langle h \rangle)$ is a *n*-ary description of an appropriate infinite limit (respectively lim, lim inf, lim sup) of *h* (if such limits exist).

Definition 2.3. For a given *n*-ary description *s* of a vector *f*, let $E_i^k(s)$ (the η-number with respect to *i*-th variable of the *k*-component) be defined as follows:

1. $E_i^1(0_n) = E_i^1(1_n) = E_i^1(\overline{1}_n) = 0$;
2. $E_i^m(c(\langle h \rangle, \langle g \rangle)) = \max_{1 \le j \le k}(E_j^m(\langle h \rangle) + E_i^j(\langle g_i \rangle))$, where *h* is a *n* components *k*-ary vector and *g* is a *k*-components *m*-ary vector;
3. for a differential recursion we distinguish two cases:
   - $i \le k$:
     $E_i^j(dr(\langle f \rangle, \langle g \rangle)) =$
     $\max(E_i^1(\langle f_1 \rangle),\ldots,E_i^1(\langle f_n \rangle), E_i^1(\langle g_1 \rangle),\ldots,E_i^1(\langle g_n \rangle),$
     $E_{k+1}^1(\langle g_1 \rangle),\ldots,E_{k+1}^1(\langle g_n \rangle))$
   - $i = k+1$
     $E_i^j(dr(\langle f \rangle, \langle g \rangle)) = \max_{0 \le m \le n}(\max(E_{k+m+1}^1(\langle g_1 \rangle),\ldots,E_{k+m+1}^1(\langle g_n \rangle)))$

   where *f* is a *n* components *k*-ary vector and *g* is a *n* components *k*+*n*+1-ary vector;
4. $E_i^k(l(\langle h \rangle)) = E_i^k(li(\langle h \rangle)) = E_i^k(ls(\langle h \rangle)) = \max(E_i^k(\langle h \rangle), E_{n+1}^k(\langle h \rangle)) + 1$, where *h* is a *k* components *n*+1-ary vector.

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

Definition 2.4. η-hierarchy is a family of $H_j = \{f : \eta(f) \le j\}$, where $E(\langle h \rangle) = \max_k \max_i E_i^k(\langle h \rangle)$ for $1 \le i \le n$, $1 \le k \le m$ and $\eta(f)$ is the minimum of $E(\langle h \rangle)$ for all possible descriptions of the function *f*.

Let us recall some propositions of real recursive functions from work [4].

Proposition 2.5. The functions +, ×, –, exp, sin, cos, $\lambda x.\frac{1}{x}$, /, ln, $\lambda xy.x^y$ are real recursive functions, and all are in H_0.

Proposition 2.6. The Kronecker δ function, the signum function, the absolute value, the Heaviside Θ function (equal to 1 if $x \ge 0$, otherwise 0), and the floor function $\lfloor x \rfloor$ are real recursive functions from H_1.

Now let us recall a mapping *GS* due to C. Moore in [6] which will be needed in further considerations. Let $\ldots a_{-2}a_{-1}.a_0a_1a_2\ldots = (a_i) = a$ be an arbitrary sequence over Σ. Let us define mapping *GS* as follows: $\Phi : a \to \partial^{F(a)}(a + G(a))$ [1], where $G : \Sigma^n \to \Sigma^n$ and $F : \Sigma^n \to \{-1, +1\}$. Here *F* is a map from *a* to the integers, and *G* is a map from *a* to finite sequences. Furthermore, we require the fact that *F* and *G* depend on a finite number of cells in *a*. We will call this area of *a* the *domain of dependence* (DOD). The notation $a + G(a)$ can be understood in the way that a finite number of cells in *a* is replaced with the sequence $G(a)$, and ∂ denotes a shift of the sequence to the left or right by the amount $F(a)$. For $F(a) = +1$ is the shift of one position to the left side direction from the dot and for $F(a) = -1$ is the shift of one position to the right side direction from the dot.

3. Simulation

The generalized shifting mapping *GS* can simulate a Turing machine. To obtain a shifting mapping *GS* which would simulate a Turing machine we should adopt a transformation method with the *G* following premise. Let us introduce a coding function of sequences over Σ which are written on the tape of the Turing machine $(\Sigma, Q, \delta, q_0, q_f)$, where $\Sigma = \{s_0, s_1, \ldots, s_m\}$ and $Q = \{q_0, q_1, \ldots q_f\}$. The symbols of Σ and of *Q* will be coded as successive digits of a coding system where $n = \max\{m, f\} + 1$ cardinality. Let $a = (a_i) = \ldots a_{-2}a_{-1}.a_0a_1a_2\ldots$ be the

[1] The function Φ has as the domain of a set of two side infinite strings over Σ and the identical counterdomain.

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

tape code TM and $a^s = \left(a_i^s\right) = \ldots a_{-2}a_{-1}.sa_0a_1a_2\ldots$ be the tape code Turing machine with the internal state Turing machine. Let $DOD\left(a_i^s\right) = a_{-1}.sa_0$ be the finite sequence $\left(a_i^s\right)$, which is the area of transformation G.

$F\left(a_i^s\right)$ – defines a movement of the head: +1 in the right side direction, -1 in the left side direction.

Definition 3.1. Let *G* be specified as follows:

$$G\left(a_i^s\right) = \begin{cases} a_{-1}.a_0's', & for\ F\left(a_i^s\right) = +1 \\ s'.a_{-1}a_0', & for\ F\left(a_i^s\right) = -1 \end{cases},$$

where a_0' (new symbol) and s' (new state) are determined by the transition function $\delta\left(a_0, s\right) \to \left(a_0', s', r\right)$ of the Turing machine respectively.

Lemma 3.2. The shifting mapping *GS* with *G* and *F* defined above simulates the Turing machine.

Example 3.3.

We present the construction of generalized shift mapping for the problem of binary successor. Let us define a Turing machine TM[2]: $(\Sigma, Q, \delta, q_0, q_f)$ as follows: $\Sigma = \{0,1,2\}$, 2 represent the empty symbol, $Q = \{0,1,2\}$, 0 is an initial state, 2 is a final state and 1 is a modification state (the Turing machine goes over type symbols from the right to the left and makes necessary changes).

The transition function δ of TM and mapping *GS* are shown in Tables 1 and 2.

Table 1. Transition function δ of Turing machine

Lp.	(Σ, Q)	$(\Sigma, Q, \{+1, -1\})$
1	(1, 0)	(1, 0, +1)
2	(0, 0)	(0, 0, +1)
3	(2, 0)	(2, 1, –1)
4	(1, 1)	(0, 1, –1)
5	(0, 1)	(1, 2, +1)
6	(2, 1)	(1, 2, –1)

[2] The sets Q and Σ have identical member, but with the different meaning: the elements of Q are states and the elements of Σ are ciphers.

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

Table 2. Mapping *GS* for δ

	Lp.	$a_{-1}.sa_0$	$G(a)$	$F(a)$
I	1	2.01	2.10	+1
	2	1.01	1.10	+1
	3	0.01	0.10	+1
II	4	2.00	2.00	+1
	5	1.00	1.00	+1
	6	0.00	1.00	+1
III	7	1.02	1.12	–1
	8	0.02	1.02	–1
IV	9	2.11	1.20	–1
	10	1.11	1.10	–1
	11	0.11	1.00	–1
V	12	2.10	2.12	+1
	13	1.10	1.12	+1
	14	0.10	0.12	+1
VI	15	2.12	2.21	–1

In the above example three different symbols can be written on a tape of TM. As during the definition of a transformation *G*, additionally we should take into account the preceding symbol with respect to the observed one on the tape, therefore we obtain three situations instead of one. That is why for a singular change of the transition function δ we should define *G* and *F* for three different arguments. The consecutive instructions of a transition function δ correspond to instructions of particular sections of mapping *GS*. Some situations on the tape are impossible. Thus in sections III and VI there are less than three instructions. For instance, section III corresponds to the third instruction in Table 1. This instruction is defined for the state 0 (the Turing machine goes over the tape the symbols from the left to the right without changing them) and observed symbol 2. In this case the appearance of 2 before the head is impossible. The above situation denotes that the input of the Turing machine is empty. Therefore we have here two instructions. Similarly in section VI we have only one possible situation.

Now let us illustrate how the mapping *GS* simulates an activity of the Turing machine. A single step will be written in the following form:

$$\ldots a_{-2}a_{-1}.sa_0a_1\ldots \overset{G}{\rightarrow} \ldots a_{-2}G(a_{-1}.sa_0)a_1\ldots \overset{F}{\rightarrow} \ldots a'_{-2}a'_{-1}.s'a'_0a'_1\ldots$$

Let an input tape of the Turing machine have the form

$$\text{T: } a_i = \ldots a_{-2}a_{-1}.a_0a_1\ldots = \ldots 222.1101222\ldots$$

then we start with: $a_i^s = \ldots a_{-2}a_{-1}.sa_0a_1\ldots = \ldots 222.01101222\ldots$

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

The successive steps of *GS* have the form:

$$\ldots 222.01101222\ldots \xrightarrow{G(2.01)} \ldots 222.10101222\ldots \xrightarrow{F(2.01)} \ldots 2221.0101222\ldots \xrightarrow{G(1.01)}$$
$$\ldots 2221.1001222\ldots \xrightarrow{F(1.01)} \ldots 22211.001222\ldots \xrightarrow{G(1.00)} \ldots 22211.001222\ldots \xrightarrow{F(1.00)}$$
$$\ldots 222110.01222\ldots \xrightarrow{G(0.01)} \ldots 222110.10222\ldots \xrightarrow{F(0.01)} \ldots 2221101.0222\ldots \xrightarrow{G(1.02)}$$
$$\ldots 2221101.1222\ldots \xrightarrow{F(1.02)} \ldots 222110.11222\ldots \xrightarrow{G(0.11)} \ldots 222111.00222\ldots \xrightarrow{F(0.11)}$$
$$\ldots 22211.100222\ldots \xrightarrow{G(1.10)} \ldots 22211.120222\ldots \xrightarrow{F(1.10)} \ldots 222111.20222\ldots$$

Therefore, for 1101 writing on the start of the Turing machine we receive number 1110.

Now let us summarize the results relating to simulating of the Turing machine by a function $R \to R$. Let us assume that the two-side infinite string: $\ldots a_{-2}a_{-1}.a_0a_1a_2\ldots$ with a base equal to *n* is transformed in to *a* right side infinite string of the form: $0.a_0a_{-1}a_1a_{-2}a_2a_{-3\ldots}$. Now we are able to assign a real number $x_a \in [0,1]$ to the above sequence.

Lemma 3.4. Let Φ be a shifting mapping of *GS*, where $DOD(a) = a_{-1}.a_0a_1$. Then there exists a function $f_{GS} : R \to R$, such that:

$$\Phi(a) = b \equiv f_{GS}(x_a) = x_b .$$

Proof. In this proof all numbers are given in the base *n*. According to the definition of *GS*, first of all we replace the elements a_{-1}, a_0 and a_1 by the elements $G(a_{|DOD})$, where $a_{|DOD} = a_{-1}.a_0a_1$, and then we shift all the digits in *a* string with respect to *F*. If *n* is the cardinality of a coding system then we have: $\lfloor x_a n^2 \rfloor - \lfloor x_a n \rfloor n = a_{-1}$, $\frac{\lfloor x_a n \rfloor}{n} = 0.a_0$, $\frac{\lfloor x_a n^3 \rfloor - \lfloor x_a n^2 \rfloor n}{n^2} = \frac{a_1}{n^2} = 0.0a_1$.

Then we have $a_{|DOD} = a_{-1}.a_0a_1 = \lfloor x_a n^2 \rfloor - \lfloor x_a n \rfloor n + \frac{\lfloor x_a n \rfloor}{n} + \frac{\lfloor x_a n^3 \rfloor - \lfloor x_a n^2 \rfloor n}{n^2}$.

Now the last string is transformed by *G* in the following way:

$$G(a_{|DOD}) = \begin{cases} a_{-1}.a_1'a_0', & for\ F(a_{|DOD}) = +1 \\ a_0'.a_{-1}a_1', & for\ F(a_{|DOD}) = -1 \end{cases}.$$

The initial string $a_{-1}.a_0a_1$ should be replaced by a new string obtained as above. So for $F(a_{|DOD}) = +1$, we obtain transformation:

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

$0.a_0a_{-1}a_1a_{-2}a_2a_{-3\ldots} \to 0.a_1'a_{-1}a_0'a_{-2}a_2a_{-3}\ldots$ and after the shift to the right we obtain: $x_b = 0.a_0'a_1'a_2a_{-1}a_3a_{-2}\ldots$. As $G\left(a_{|DOD}\right)n^2 = a_{-1}a_1'a_0'$ and $\left\lfloor G\left(a_{|DOD}\right)n\right\rfloor n = a_{-1}a_1'0$ [3], then $\dfrac{G\left(a_{|DOD}\right)n^2 - \left\lfloor G\left(a_{|DOD}\right)n\right\rfloor n}{n} = 0a_0$. Continuing the process we have $\left\lfloor G\left(a_{|DOD}\right)n\right\rfloor = a_{-1}a_1'$ and $\left\lfloor G\left(a_{|DOD}\right)\right\rfloor n = a_{-1}0$, analogously, as before, we have $\dfrac{\left\lfloor G\left(a_{|DOD}\right)n\right\rfloor - \left\lfloor G\left(a_{|DOD}\right)\right\rfloor n}{n^2} = 0.0a_1'$. If we compute the sum of the above strings, then we obtain the first three elements of x_b, i.e. $0.a_0'a_1'$. The successive elements are shifted twice: all even elements to the right and all odds to the left. Hence the string $0.000a_{-1}0a_{-2}0a_{-3\ldots}$ is expressed by a series $\sum_{k=1}^{\infty}\dfrac{\left\lfloor x_a n^{2k}\right\rfloor - \left\lfloor x_a n^{2k-1}\right\rfloor n}{n^{2k+2}}$ and the second string $0.00a_20a_30a_4\ldots$ by $\sum_{k=1}^{\infty}\dfrac{\left\lfloor x_a n^{2k+3}\right\rfloor - \left\lfloor x_a n^{2k+2}\right\rfloor n}{n^{2k+1}}$. Therefore the final formula has the form:

$$f_{GS}\left(x_a\right) = \frac{G\left(a_{|DOD}\right)n^2 - \left\lfloor G\left(a_{|DOD}\right)n\right\rfloor n}{n} + \frac{\left\lfloor G\left(a_{|DOD}\right)n\right\rfloor - \left\lfloor G\left(a_{|DOD}\right)\right\rfloor n}{n^2}$$
$$+ \lim_{m\to\infty}\sum_{k=1}^{m}\frac{\left\lfloor x_a n^{2k}\right\rfloor - \left\lfloor x_a n^{2k-1}\right\rfloor n}{n^{2k+2}} + \lim_{m\to\infty}\sum_{k=1}^{m}\frac{\left\lfloor x_a n^{2k+3}\right\rfloor - \left\lfloor x_a n^{2k+2}\right\rfloor n}{n^{2k+1}}.$$

For $F\left(a_{|DOD}\right) = -1$ we obtain the similar formula:

$$f_{GS}\left(x_a\right) = \frac{\left\lfloor G\left(a_{|DOD}\right)\right\rfloor}{n} + \frac{G\left(a_{|DOD}\right)n^2 - \left\lfloor G\left(a_{|DOD}\right)n\right\rfloor n}{n^5} + \frac{\left\lfloor x_a n^2\right\rfloor - \left\lfloor x_a n\right\rfloor n}{n^3}$$
$$+ \lim_{m\to\infty}\sum_{k=1}^{m}\frac{\left\lfloor x_a n^{2k+3}\right\rfloor - \left\lfloor x_a n^{2k+2}\right\rfloor n}{n^{2k+5}} + \lim_{m\to\infty}\sum_{k=1}^{m}\frac{\left\lfloor x_a n^{2k+2}\right\rfloor - \left\lfloor x_a n^{2k+1}\right\rfloor n}{n^{2k}}$$

□

This result can be used for a formulation of the following theorem:

Theorem 3.5. For an arbitrary Turing machine TM: there exists the real recursive function $f_{GS}: R \to R$ belonging to H_2, that simulates this Turing machine.

[3] where $a_{-1}a_1'0$ denotes the number with digits a_{-1}, a_1', 0 in the base n.

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

Proof. It is shown in [4] that addition, subtraction, multiplication and power are the elements of H_0, whenever $\lfloor x \rfloor$ is in H_1. Defining $f_{GS}(x_a) = x_b$ we use a composition of two operations $\lfloor x \rfloor$; we have $\lfloor G(a_{|DOD}) \rfloor$ which is in H_2 because $a_{|DOD} = \lfloor yn \rfloor + \frac{\lfloor xn^2 \rfloor}{n^2}$ is in H_1. The above limits exist, and they are finite and belong to H_2. For instance, we consider $\lim_{m\to\infty} \sum_{k=1}^{m} \frac{\lfloor x_a n^{2k} \rfloor - \lfloor x_a n^{2k-1} \rfloor n}{n^{2k+2}} = \sum_{k=1}^{\infty} \frac{\lfloor x_a n^{2k} \rfloor - \lfloor x_a n^{2k-1} \rfloor n}{n^{2k+2}}$.

This series $\sum_{k=1}^{\infty} \frac{\lfloor x_a n^{2k} \rfloor - \lfloor x_a n^{2k-1} \rfloor n}{n^{2k+2}}$ is convergent because it is bounded by the corresponding sum $\sum_{k=1}^{\infty} \frac{n-1}{n^{2k+2}}$. From the Weierstrass' theorem the considered series is convergent. Therefore $\lim_{m\to\infty} \sum_{k=1}^{m} \frac{\lfloor x_a n^{2k} \rfloor - \lfloor x_a n^{2k-1} \rfloor n}{n^{2k+2}}$ by definition 2.3 is in class H_2. The other limits behave similarly.

□

Now we show the result of the simulation of the Turing machine by two-dimensional real recursive function $R^2 \to R^2$. We represent a two-side infinite sequence: $\ldots a_{-2}a_{-1}.a_0a_1a_2\ldots$ by the pair of infinite sequences: $(0.a_0a_1a_2\ldots, 0.a_{-1}a_{-2}a\ldots)$.

Lemma 3.6. Let Φ be a shifting mapping of *GS* with $DOD(a) = a_{-1}.a_0a_1$. Then there exists a function $f_{GS} : R^2 \to R^2$, such that:

$$\Phi(a) = b \equiv f_{GS}(x_a, y_a) = (x_b, y_b).$$

Proof. According to the definition of *GS*, (x_b, y_b) is obtained from (x_a, y_a) by replacing a_{-1}, a_0 and a_1 by the elements of $G(a_{|DOD})$, where $a_{|DOD} = a_{-1}.a_0a_1$ and then by shifting with respect to F. Multiplying y_a by n, $y_a n$ and taking $\lfloor y_a n \rfloor$, we obtain a_{-1}. Analogously we have $\frac{\lfloor x_a n^2 \rfloor}{n^2} = 0.a_0a_1$, hence $a_{|DOD} = a_{-1}.a_0a_1 = \lfloor y_a n \rfloor + \frac{\lfloor x_a n^2 \rfloor}{n^2}$. Then the last string is transformed by *G*. For

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

$F\left(a_{|DOD}\right)=+1$ the elements a_0a_1 should be replaced by $a_1'a_0'$, whereas in a string y_a symbol a_{-1} remains the same.

Hence we have: $\left(0.a_0a_1a_2\ldots,0.a_{-1}a_{-2}a_{-3}\ldots\right)\rightarrow\left(0.a_1'a_0'a_2\ldots,0.a_{-1}a_{-2}a_{-3}\ldots\right)$ and after shift we obtain: $\left(x_b,y_b\right)=\left(0.a_0'a_2a_3\ldots,0.a_1'a_{-1}a_{-2}\ldots\right)$. Multiplying $\left\lfloor G\left(a_{|DOD}\right)n\right\rfloor=a_{-1}a_1'$ by n and subtracting from $G\left(a_{|DOD}\right)n^2=a_{-1}a_1'a_0'$ we obtain a_0', $\dfrac{G\left(a_{|DOD}\right)n^2-\left\lfloor G\left(a_{|DOD}\right)n\right\rfloor n}{n}=0.a_0'$. Thus we have:

$$x_b=\frac{G\left(a_{|DOD}\right)n^2-\left\lfloor G\left(a_{|DOD}\right)n\right\rfloor n}{n}+\frac{x_an^2-\left\lfloor x_an^2\right\rfloor}{n}\ .$$

Continuing similar considerations we have $\left\lfloor G\left(a_{|DOD}\right)n\right\rfloor=a_{-1}a_1'$ and $\left\lfloor G\left(a_{|DOD}\right)\right\rfloor=a_{-1}$. From $\dfrac{\left\lfloor G\left(a_{|DOD}\right)n\right\rfloor-\left\lfloor G\left(a_{|DOD}\right)\right\rfloor n}{n}$ we obtain $0.a_1'$. A desired string y_b is equal to:

$$\frac{\left\lfloor G\left(a_{|DOD}\right)n\right\rfloor-\left\lfloor G\left(a_{|DOD}\right)\right\rfloor n}{n}+\frac{y_a}{n}\ .$$

By parallel considerations as above for $F\left(a_{|DOD}\right)=-1$ we find:

$$x_b=\frac{\left\lfloor G\left(a_{|DOD}\right)\right\rfloor}{n}+\frac{\left\lfloor y_an\right\rfloor}{n^2}+\frac{G\left(a_{|DOD}\right)n^2-\left\lfloor G\left(a_{|DOD}\right)n\right\rfloor n}{n^3}+\frac{x_an^3-\left\lfloor x_an^3\right\rfloor}{n^3}$$

and

$$y_b=y_an-\left\lfloor y_an\right\rfloor$$

□

The above result allows to formulate the following theorem.

Theorem 3.7. For an arbitrary Turing machine TM: $(\Sigma,Q,\delta,q_0,q_f)$, there exists a recursive function $f_{GS}:R^2\rightarrow R^2$ belonging to H_2, that simulates this Turing machine.

Proof. Defining $f_{GS}\left(x_a,y_a\right)=\left(x_b,y_b\right)$ we use a composition of two operations $\lfloor x\rfloor$; we have $\left\lfloor G\left(a_{|DOD}\right)\right\rfloor$ where $a_{|DOD}=\left\lfloor y_an\right\rfloor+\dfrac{\left\lfloor x_an^2\right\rfloor}{n^2}$. Hence f_{GS} is in H_2.

□

It is an interesting phenomenon if we increase dimension of simulation from 1 to 2, the class of the functions in η-hierarchy is not changed.

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

Now we propose the next simulation of the Turing machine by a real recursive function $R^5 \to R^5$. We represent a two-side infinite sequence: $\ldots a_{-2}a_{-1}.a_0a_1a_2\ldots$ as five elements (two infinite sequences and three digits) as follows: $\left(0.a_2a_3a_4\ldots, 0.a_{-2}a_{-3}a_{-4}\ldots, a_{-1}, a_0, a_1\right)$. The three last digits are respectively: a_{-1} – the first digit before dot (symbol directly before that under the head), a_0 – the first digit after dot (actually state machine), a_1 – the second digit after dot (a symbol under the head), however, two infinite sequences are the right and left parts of the sequence respectively.

Lemma 3.8. Let Φ be a shifting mapping of *GS* with $DOD(a) = a_{-1}.a_0a_1$. Then there exists a function $f_{GS} : R^5 \to R^5$, such that:

$$\Phi(a) = b \equiv f_{GS}(x_a, y_a, p_a, q_a, s_a) = (x_b, y_b, p_b, q_b, s_b).$$

Proof. Analogously to lemma 3.6, the function f_{GS} should replace a_{-1}, a_0 and a_1 by the digits determined by $G\left(a_{|DOD}\right)$, where $a_{|DOD} = a_{-1}.a_0a_1$, and then the shift digits according to the value of F. In this case $a_{|DOD}$ can be computed in an easier way: $a_{|DOD} = p_a + \frac{q_a}{n} + \frac{s_a}{n^2}$. Let us consider at the beginning the case $F\left(a_{|DOD}\right) = +1$. In this case a transformation *G* changes only the old value q_a into $a_1^{'}$ and the old value s_a into $a_0^{'}$. Since in this case transformation *G* does not influence digit a_{-1} (changes neither value nor position) a variable p_a does not change.

$$\left(0.a_2a_3a_4\ldots, 0.a_{-2}a_{-3}a_{-4}\ldots, a_{-1}, a_0, a_1\right) \to \left(0.a_2a_3a_4\ldots, 0.a_{-2}a_{-3}a_{-4}\ldots, a_{-1}, a_1^{'}, a_0^{'}\right).$$

By shift to the right we obtain the following digits:

$$\left(0.a_2a_3a_4\ldots, 0.a_{-2}a_{-3}a_{-4}\ldots, a_{-1}, a_1^{'}, a_0^{'}\right) \to \left(0.a_3a_4a_5\ldots, 0.a_{-1}a_{-2}a_{-3}\ldots, a_1^{'}, a_0^{'}, a_2\right).$$

Hence

$$\left(x_b, y_b, p_b, q_b, s_b\right) = \left(0.a_3a_4a_5\ldots, 0.a_{-1}a_{-2}a_{-3}\ldots, a_1^{'}, a_0^{'}, a_2\right).$$

We have:

$$x_b = x_a n - \lfloor x_a n \rfloor = a_2.a_3a_4a_5\ldots - a_2 = 0.a_3a_4a_5\ldots,$$

$$y_b = \frac{p_a}{n} + \frac{y_a}{n} = 0.a_{-1} + 0.0a_{-2}a_{-3}\ldots = 0.a_{-1}a_{-2}a_{-3}\ldots,$$

$$p_b = \left\lfloor G\left(a_{|DOD}\right)n \right\rfloor - \left\lfloor G\left(a_{|DOD}\right) \right\rfloor n = a_{-1}a_1^{'} - a_{-1}0 = a_1^{'},$$

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

$$q_b = G\left(a_{|DOD}\right)n^2 - \left\lfloor G\left(a_{|DOD}\right)n \right\rfloor n = a_{-1}a_1'a_0' - a_{-1}a_1'0 = a_0',$$
$$s_b = \left\lfloor x_a n \right\rfloor = a_2.$$

In the case $F\left(a_{|DOD}\right) = -1$ the transformation G changes the old value p_a for the a_0', the old value q_a for the a_{-1} and the old value s_a for the a_1', and we have:

We obtain the mapping:

$$\left(0.a_2a_3a_4\ldots, 0.a_{-2}a_{-3}a_{-4}\ldots, a_{-1}, a_0, a_1\right) \to \left(0.a_2a_3a_4\ldots, 0.a_{-2}a_{-3}a_{-4}\ldots, a_0', a_{-1}, a_1'\right).$$

Shifting in the left hand side direction we obtain the following motion digits:

$$\left(0.a_2a_3a_4\ldots, 0.a_{-2}a_{-3}a_{-4}\ldots, a_0', a_{-1}, a_1'\right) \to \left(0.a_1'a_2a_3\ldots, 0.a_{-3}a_{-4}a_{-5}\ldots, a_{-2}, a_0', a_{-1}\right).$$

Hence

$$\left(x_b, y_b, p_b, q_b, s_b\right) = \left(0.a_1'a_2a_3\ldots, 0.a_{-3}a_{-4}a_{-5}\ldots, a_{-2}, a_0', a_{-1}\right).$$

So we have:

$$x_b = \frac{G\left(a_{|DOD}\right)n^2 - \left\lfloor G\left(a_{|DOD}\right)n \right\rfloor n}{n} + \frac{x_a}{n} = 0.a_1' + 0.0a_2a_3a_4\ldots = 0.a_1'a_2a_3\ldots$$

for $G\left(a_{|DOD}\right)n^2 - \left\lfloor G\left(a_{|DOD}\right)n \right\rfloor n = a_0'a_{-1}a_1' - a_0'a_{-1}0 = a_1'$,

$$y_b = y_a n - \left\lfloor y_a n \right\rfloor = a_{-2}.a_{-3}a_{-4}\ldots - a_{-2} = 0.a_{-3}a_{-4}a_{-5}\ldots,$$
$$p_b = \left\lfloor y_a n \right\rfloor = a_{-2},$$
$$q_b = \left\lfloor G\left(a_{|DOD}\right) \right\rfloor = a_0',$$
$$s_b = p_a = a_{-1}.$$

□

The above result allows to formulate the following theorem.

Theorem 3.9. For an arbitrary Turing machine TM: $(\Sigma, Q, \delta, q_0, q_f)$ there exists the real recursive function $f_{GS} : R^5 \to R^5$ belonging to H_1, that simulates this Turing machine.

Proof. Let us observe that $a_{|DOD} = p_a + \frac{q_a}{n} + \frac{s_a}{n^2}$ is in H_0.

Defining $f_{GS}\left(x_a, y_a, p_a, q_a, s_a\right) = \left(x_b, y_b, p_b, q_b, s_b\right)$ we use the operation $\lfloor x \rfloor$, therefore f_{GS} is in H_1.

□

To define the function $f_{GS} : R^5 \to R^5$ three different variables p_a, q_a and s_a (each of these symbols is one of the digits occurring in $a_{|DOD}$) have been used.

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 28/11/2026 02:10:18

This allows to define $a_{|DOD}$ without a function $\lfloor x \rfloor$. This fact makes it possible to decrease the class of functions f_{GS} in η-hierarchy.

4. Conclusions

This paper deals with complete description of shifting transformation *GS* proposed by C. Moore in [5]. Three simulation algorithms of the Turing machines by using real recursive functions, with the extended shifting mappings to the set of real functions are given. We determine the position in the η-hierarchy for these simulations. However, one can simulate Turing machines without using the shifting mapping. Such a method has been used in work [9]. One and two-dimensional simulations are from H_2, while the simulation with 5 arguments is H_1. We treat these results as the first step in the research of possible simulations of different types of Turing machines by real recursive functions. The next step will be devoted to accelerating Turing machines and O – machines.

Acknowledgements

I would like to thank Jerzy Mycka for suggesting the problem and for helpful discussions. I am grateful to Zdzisław Grodzki for careful reading of this paper.

References

[1] Ord T., *Hypercomputation: computing more than the Turing machine*, Honours Thesis, University of Melbourne, (2002).
[2] Copeland B.J., *Even Turing machines can compute uncomputable functions*. In: C.S. Calude, J. Casti, and M.J. Dinneen (eds), Unconventional Models of Computation, Springer-Verlag, (1998).
[3] Lokhorst Gert-Jan C., *Hypercomputation,* Department of Philosophy, University of Helsinki, (2001).
[4] Mycka J., Costa J.F., *Real recursive functions and their hierarchy*, Submitted to Journall of Complexity.
[5] Moore C., *Recursion theory on the reals and continuous-time computation*, Theoretical Computer Science, 162 (1996) 23.
[6] Moore C., *Unpredictability and Undecidability in Dynamical Systems*, Physical Review Letters, 64(20) (1990) 2354.
[7] Odifreddi P., *Classical recursion theory*, Elsevier, (1989).
[8] Odifreddi P., *Classical recursion theory II*, Elsevier, (1999).
[9] Koiran P., Moore C., *Closed-form analytic maps in one and two dimensions can simulate universal Turing machines*, Theoretical Computer Science 210 (1999) 217.