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 $G_S$ is used to this end. The relationship between a simulation dimension and classes of $\eta$-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 $\lambda$ – 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
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 $T_M$ 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, $\delta : \Sigma \times Q \rightarrow \Sigma \times Q \times \{-1, +1\}$ 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: $\alpha(q, \alpha_2)$, where $q \in Q$ is the current state $T_M$, and $\alpha_2$ is the two-side infinite string written in the tape. We suppose that the head observes the first from the symbol of the string $\alpha_2$.

We define the Turing machines work as follows way. Let $\ldots x_1x_2 \ldots x_{i-1}qx_i \ldots x_n \ldots$ be the Turing machines temporary description. For $\delta(q, x_i) = (p, y, -1)$ we have: $\ldots x_1x_2 \ldots x_{i-1}qx_i \ldots x_n \ldots \rightarrow \ldots x_1x_2 \ldots x_{i-1}px_i y \ldots x_n \ldots$; however for $\delta(q, x_i) = (p, y, +1)$ we have: $\ldots x_1x_2 \ldots x_{i-1}qx_i \ldots x_n \ldots \rightarrow \ldots x_1x_2 \ldots x_{i-1}yp \ldots x_n \ldots$.

Now let us recall the basic definition relating to the recursive functions over reals. In particular, we describe $\eta$-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 $(1 \leq i \leq n)$

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

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,
then the vector \( h \) of \( n \) \( k+1 \)-ary components which is the solution of the Cauchy problem for \( 1 \leq i \leq n \)
\[
h_i(x_1, \ldots, x_k, 0) = f_i(x_1, \ldots, x_k),
\]
\[
\partial_y h_i(x_1, \ldots, x_k, y) = g_i(x_1, \ldots, x_k, y, h_1(x_1, \ldots, x_k, y), \ldots, h_n(x_1, \ldots, x_k, y))
\]
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
\[
h_i(x_1, \ldots, x_k) = \lim_{y \to \infty} f_i(x_1, \ldots, x_k, y),
\]
\[
h'_i(x_1, \ldots, x_k) = \liminf_{y \to \infty} f_i(x_1, \ldots, x_k, y),
\]
\[
h''_i(x_1, \ldots, x_k) = \limsup_{y \to \infty} f_i(x_1, \ldots, x_k, y)
\]
are real recursive, whenever these limits are defined for all \( 1 \leq i \leq n \).

4. Arbitrary real recursive vectors can be 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 \( \eta \)-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: \( i'_k \) as a \( k \)-ary description for projection \( I'_k \) for all \( 1 \leq i \leq k \); \( 1_k, \overline{1}_k, 0_k \) are \( k \)-ary descriptions for constants 1, -1, 0 used with \( k \) variables. We must add also operator symbols (descriptors) for all introduced
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: \( i_1, i_2, \overline{1}_n, 0_n \) are \( n \)-ary descriptions of \( I_n^j, \) \( 1 \leq j \leq n \in \mathbb{N}, \) \( \lambda x_1, \ldots, x_n.1, \) \( \lambda x_1, \ldots, x_n.0 \) for all \( (x_1, \ldots, x_n) \in \mathbb{R}^n, \) \( n \in \mathbb{N}, \) 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 \( l(\langle h \rangle), l_1(\langle h \rangle), l_2(\langle h \rangle) \) 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 \( \eta \)-number with respect to \( i \)-th variable of the \( k \)-component) be defined as follows:

1. \( E^i_k(0_n) = E^i_k(1_n) = E^i_k(\overline{1}_n) = 0 \);
2. \( E^i_k(c(\langle h \rangle, \langle g \rangle)) = \max_{1 \leq j \leq k} (E^i_j(\langle h \rangle) + E^i_j(\langle g \rangle)) \), 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:
   - \( i \leq k \):
     \[
     E^i_k(dr(\langle f \rangle, \langle g \rangle)) = \max(E^i_k(\langle f_1 \rangle), \ldots, E^i_k(\langle f_n \rangle), E^i_k(\langle g_1 \rangle), \ldots, E^i_k(\langle g_n \rangle), \\
     E^i_{k+1}(\langle g_1 \rangle), \ldots, E^i_{k+1}(\langle g_n \rangle))
     \]
   - \( i = k+1 \):
     \[
     E^i_k(dr(\langle f \rangle, \langle g \rangle)) = \max_{0 \leq m \leq n} (\max(E^i_{k+m+1}(\langle g_1 \rangle), \ldots, E^i_{k+m+1}(\langle g_n \rangle)))
     \]
   where \( f \) is a \( n \)-components \( k \)-ary vector and \( g \) is a \( n \)-components \( k+n+1 \)-ary vector;
4. \( E^i_k(l(\langle h \rangle)) = E^k_i(l(\langle h \rangle)) = E^k_i(l_1(\langle h \rangle)) = \max(E^k_i(\langle h \rangle), E^k_{n+1}(\langle h \rangle)) + 1 \),
   where \( h \) is a \( k \)-components \( n+1 \)-ary vector.
Definition 2.4. \( \eta \)-hierarchy is a family of \( H_j = \{ f : \eta(f) \leq j \} \), where \( E(h) = \max_k \max_i E_i^k(h) \) for \( 1 \leq i \leq n \), \( 1 \leq k \leq m \) and \( \eta(f) \) is the minimum of \( E(h) \) 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 +, \( \times \), \(-\), exp, sin, cos, \( \lambda x.\frac{1}{x} \), ln, \( \lambda x.y.x^y \) are real recursive functions, and all are in \( H_0 \).

Proposition 2.6. The Kronecker \( \delta \) function, the signum function, the absolute value, the Heaviside \( \Theta \) function (equal to 1 if \( x \geq 0 \), 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 \( \Sigma \). Let us define mapping \( GS \) as follows: \( \Phi : a \rightarrow \delta^{F(a)}(a + G(a)) \), where \( G : \Sigma^n \rightarrow \Sigma^n \) and \( F : \Sigma^n \rightarrow \{-1,+1\} \). 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 \( \delta \) 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 \( \Sigma \) 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 \( \Sigma \) 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 \( \Phi \) has as the domain of a set of two side infinite strings over \( \Sigma \) and the identical counterdomain.
tape code TM and \( a_i^t = (a_i^t) = \ldots a_{-2} a_{-1} s a_0 a_1 a_2 \ldots \) be the tape code Turing machine with the internal state Turing machine. Let \( DOD(a_i^t) = a_{-1} s a_0 \) be the finite sequence \( (a_i^t) \), which is the area of transformation \( G \).

\( F(a_i^t) \) – 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:

\[
G(a_i^t) = \begin{cases}
  a_{-1} a_0 a_i, & \text{for } F(a_i^t) = +1 \\
  s a_{-1} a_0, & \text{for } F(a_i^t) = -1,
\end{cases}
\]

where \( a_0' \) (new symbol) and \( s' \) (new state) are determined by the transition function \( \delta(a_0, s) \rightarrow (a_0', s', r) \) 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 \( \delta \) of TM and mapping \( GS \) are shown in Tables 1 and 2.

<table>
<thead>
<tr>
<th>Lp.</th>
<th>((\Sigma,Q))</th>
<th>((\Sigma,Q,{+1,-1}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1, 0)</td>
<td>(1, 0, +1)</td>
</tr>
<tr>
<td>2</td>
<td>(0, 0)</td>
<td>(0, 0, +1)</td>
</tr>
<tr>
<td>3</td>
<td>(2, 0)</td>
<td>(2, 1, -1)</td>
</tr>
<tr>
<td>4</td>
<td>(1, 1)</td>
<td>(0, 1, -1)</td>
</tr>
<tr>
<td>5</td>
<td>(0, 1)</td>
<td>(1, 2, +1)</td>
</tr>
<tr>
<td>6</td>
<td>(2, 1)</td>
<td>(1, 2, -1)</td>
</tr>
</tbody>
</table>

\(^2\) The sets \( Q \) and \( \Sigma \) have identical member, but with the different meaning: the elements of \( Q \) are states and the elements of \( \Sigma \) are ciphers.
Three simulations of Turing machines with the use ...

Table 2. Mapping $GS$ for $\delta$

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

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 $\delta$ we should define $G$ and $F$ for three different arguments. The consecutive instructions of a transition function $\delta$ 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:

$$
\ldots a_{-2}a_{-1}sa_0a_1 \ldots \rightarrow \ldots a_{-2}G(a_{-1}sa_0)a_1 \ldots \rightarrow \ldots a_{-2}a_{-1}sa_0a_1 \ldots
$$

Let an input tape of the Turing machine have the form

$$
T: a_i = \ldots a_{-2}a_{-1}a_0a_1 = \ldots 222.11011222 \ldots
$$

then we start with: $a_i^r = \ldots a_{-2}a_{-1}sa_0a_1 = \ldots 222.011010222 \ldots$
The successive steps of GS have the form:

\[
\begin{align*}
\text{Step 1:} & \quad 2.01 2.01 1.01 \\
\text{Step 2:} & \quad 1.01 1.00 1.00 \\
\text{Step 3:} & \quad 0.01 0.01 1.02 \\
\end{align*}
\]

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 \( R \rightarrow R \). Let us assume that the two-side infinite string: \( \ldots a_2 a_{-1} a_0 a_1 a_2 \ldots \) with a base equal to \( n \) is transformed in to a right side infinite string of the form: \( 0.a_0 a_{-1} a_2 a_3 \ldots \). Now we are able to assign a real number \( x_a \in [0,1] \) to the above sequence.

**Lemma 3.4.** Let \( \Phi \) be a shifting mapping of GS, where \( DOD(a) = a_{-1} a_0 a_1 \).

Then there exists a function \( f_{GS} : R \rightarrow R \), 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 \( a_{DOD} = a_{-1} a_0 a_1 \), 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:

\[
\left\lfloor \frac{x_n^2}{n} \right\rfloor - \left\lfloor \frac{x_n}{n} \right\rfloor n = a_{-1}, \quad \frac{x_n}{n} = 0.a_0, \quad \frac{\left\lfloor x_n^3 - \left\lfloor x_n^2 \right\rfloor n \right\rfloor}{n^2} = a_1 = 0.a_1.
\]

Then we have \( a_{DOD} = a_{-1} a_0 a_1 = \left\lfloor \frac{x_n^2}{n} \right\rfloor - \left\lfloor \frac{x_n}{n} \right\rfloor n + \frac{x_n}{n} + \frac{\left\lfloor x_n^3 - \left\lfloor x_n^2 \right\rfloor n \right\rfloor}{n^2} \).

Now the last string is transformed by \( G \) in the following way:

\[
G\left(a_{DOD}\right) = \begin{cases} 
\frac{a_{-1} a_1 \cdot a_0}{a_1}, & \text{for } F\left(a_{DOD}\right) = +1 \\
\frac{a_0 a_{-1} a_1}{a_1}, & \text{for } F\left(a_{DOD}\right) = -1 
\end{cases}
\]

The initial string \( a_{-1} a_0 a_1 \) should be replaced by a new string obtained as above.

So for \( F\left(a_{DOD}\right) = +1 \), we obtain transformation:
Three simulations of Turing machines with the use ...

0.a_0a_{-1}a_1a_{-2}a_2a_{-3}... \rightarrow 0.a'_0a_{-1}a'_1a_{-2}a_2a_{-3}... and after the shift to the right we obtain: $x_b = 0.a'_0a_{-1}a'_1a_{-2}a_2...$. As $G(a_{DOD})n^2 = a_{-1}a'_1a_0$ and $\left\lfloor G(a_{DOD})n \right\rfloor n = a_{-1}a'_1$\footnote{where $a_{-1}a'_1$ denotes the number with digits $a_{-1}, a'_1, 0$ in the base $n$.}, then $G(a_{DOD})n^2 - \left\lfloor G(a_{DOD})n \right\rfloor n = 0a_0$. Continuing the process we have $\left\lfloor G(a_{DOD})n \right\rfloor n = a_{-1}a'_1$ and $\left\lfloor G(a_{DOD})n \right\rfloor n = a_{-1}0$, analogously, as before, we have $\left\lfloor G(a_{DOD})n \right\rfloor n^2 - \left\lfloor G(a_{DOD})n \right\rfloor n = 0.0a'_1$. If we compute the sum of the above strings, then we obtain the first three elements of $x_b$, i.e. $0.a_0a'_1$. The successive elements are shifted twice: all even elements to the right and all odds to the left. Hence the string $0.000a_{-1}a_{-2}a_{-3}...$ is expressed by a series

$$\sum_{k=1}^{\infty} \frac{x_an^{2k}}{n^{2k+2}} - \frac{x_an^{2k-1}}{n^{2k+1}} n$$

and the second string $0.00a_{-2}a_{-3}a_{-4}...$ by

$$\sum_{k=1}^{\infty} \frac{x_an^{2k+3}}{n^{2k+5}} - \frac{x_an^{2k+2}}{n^{2k+4}} n$$

Therefore the final formula has the form:

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

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

$$f_{GS}(x_a) = \frac{G(a_{DOD})}{n} + \frac{G(a_{DOD})n^2 - \left\lfloor G(a_{DOD})n \right\rfloor n}{n^2} + \frac{x_an^2}{n^3} - \frac{x_an^2}{n^3} n$$

$$+ \lim_{m \to \infty} \sum_{k=1}^{m} \frac{x_an^{2k+3}}{n^{2k+5}} - \frac{x_an^{2k+2}}{n^{2k+5}} n$$

The 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.
Proof. It is shown in [4] that addition, subtraction, multiplication and power are the elements of $H_1$, 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 $a_{DOD} = \lfloor yn \rfloor + \lfloor \frac{xn^2}{n^2} \rfloor$ 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: $\ldots a_2a_1a_0a_1a_2 \ldots$ by the pair of infinite sequences: $(0.a_0a_1a_2 \ldots, 0.a_1a_2a_3 \ldots)$.

Lemma 3.6. Let $\Phi$ 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 $\lfloor \frac{x_a n^2}{n^2} \rfloor = 0.a_0a_1$, hence $a_{DOD} = a_{-1}a_0a_1 = \lfloor y_a n \rfloor + \lfloor \frac{x_a n^2}{n^2} \rfloor$. Then the last string is transformed by $G$. For
Three simulations of Turing machines with the use ...

\[ F(a_{\text{DOD}}) = 1 \]

the elements \(a_0a_i\) should be replaced by \(a_0^i a_i\), whereas in a string \(y_a\) symbol \(a_{-1}\) remains the same.

Hence we have: \((0.a_0a_1a_2\ldots,0.a_{-1}a_{-2}a_{-3}\ldots) \rightarrow (0.a_0^i a_1 a_2\ldots,0.a_0^i a_{-1} a_{-2} a_{-3}\ldots)\) and after shift we obtain: \((x_b, y_b) = (0.a_0^i a_2 a_3\ldots,0.a_0^i a_{-1} a_{-2}\ldots)\). Multiplying \(G(a_{\text{DOD}})n^2 = a_{-1}a_0^i\) by \(n\) and subtracting from \(G(a_{\text{DOD}})n^2 = a_{-1}a_0^i\) we obtain

\[
G(a_{\text{DOD}})n^2 - G(a_{\text{DOD}})n \left\lfloor \frac{n}{n} \right\rfloor = 0a_0^i.
\]

Thus we have:

\[
x_b = \frac{G(a_{\text{DOD}})n^2 - G(a_{\text{DOD}})n \left\lfloor \frac{n}{n} \right\rfloor}{n} + \frac{x_2n^3 - \left\lfloor x_2n^3 \right\rfloor}{n^3}.
\]

Continuing similar considerations we have \(G(a_{\text{DOD}})n = a_{-1}a_0^i\) and \(G(a_{\text{DOD}}) = a_{-1}\). From

\[
\left\lfloor G(a_{\text{DOD}})n \right\rfloor - \left\lceil G(a_{\text{DOD}})n \right\rceil \frac{n}{n} + y_a\]

we obtain \(0a_0^i\). A desired string \(y_b\) is equal to:

\[
\left\lfloor G(a_{\text{DOD}})n \right\rfloor - \left\lceil G(a_{\text{DOD}})n \right\rceil \frac{n}{n} + y_a\]

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

\[
x_b = \frac{\left\lfloor G(a_{\text{DOD}})n \right\rfloor + \left\lfloor y_a n \right\rfloor}{n} + \frac{G(a_{\text{DOD}})n^2 - G(a_{\text{DOD}})n \left\lfloor \frac{n}{n} \right\rfloor}{n^3} + \frac{x_2n^3 - \left\lfloor x_2n^3 \right\rfloor}{n^3}
\]

and

\[
y_b = y_a n - \left\lfloor y_a n \right\rfloor.
\]

The above result allows to formulate the following theorem.

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

**Proof.** Defining \(f_{GS}(x_a, y_a) = (x_b, y_b)\) we use a composition of two operations \(\left\lfloor x \right\rfloor\); we have \(G(a_{\text{DOD}})n = \left\lfloor y_a n \right\rfloor + \frac{\left\lfloor x_2 n^3 \right\rfloor}{n^3}\). 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 \(\eta\)-hierarchy is not changed.
Now we propose the next simulation of the Turing machine by a real recursive function $R^5 \rightarrow R^5$. 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: $(0.a_2a_3a_4 \ldots, 0.a_{-2}a_{-3}a_{-4} \ldots, a_{-1}, a_0, a_1)$. 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 $\Phi$ be a shifting mapping of $GS$ with $DOD(a) = a_{-1}a_0a_1$. Then there exists a function $f_{GS}: R^5 \rightarrow R^5$, 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 $G(a_{DOD})$, 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: $a_{DOD} = p_a + q_a + s_a$. Let us consider at the beginning the case $F(a_{DOD}) = +1$. In this case a transformation $G$ changes only the old value $q_a$ into $a_1\prime$ and the old value $s_a$ into $a_0\prime$. Since in this case transformation $G$ does not influence digit $a_{-1}$ (changes neither value nor position) a variable $p_a$ does not change.

$$(0.a_2a_3a_4 \ldots, 0.a_{-2}a_{-3}a_{-4} \ldots, a_{-1}, a_0, a_1) \rightarrow (0.a_2a_3a_4 \ldots, 0.a_{-2}a_{-3}a_{-4} \ldots, a_{-1}, a_1\prime, a_0\prime).$$

By shift to the right we obtain the following digits:

$$(0.a_2a_3a_4 \ldots, 0.a_{-2}a_{-3}a_{-4} \ldots, a_{-1}, a_1, a_0) \rightarrow (0.a_3a_4a_5 \ldots, 0.a_{-1}a_{-2}a_{-3} \ldots, a_1\prime, a_0\prime, a_2).$$

Hence

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

We have:

$$x_b = x_a n - \left[ x_a n \right] = a_2.a_3a_4a_5 \ldots - a_2 = 0.a_3a_4a_5 \ldots,$$

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

$$p_b = \left[ G(a_{DOD}) \right] n - \left[ G(a_{DOD}) \right] n = a_{-1}a_{-1} - a_{-1}0 = a_{-1}.$$


Three simulations of Turing machines with the use ...

\[ q_b = G(a_{i_DOD}) n^2 - G(a_{i_DOD}) n = a_{-1} a_i a_0 - a_{-1} a'_i 0 = a'_i , \]

\[ s_b = \left\lfloor x_a n \right\rfloor = a_2 . \]

In the case \( F(a_{i_DOD}) = -1 \) the transformation \( G \) changes the old value \( p_a \) for the \( a'_i \), 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:

\( (0.a_2 a_3 a_4 \ldots, 0.a_{-2} a_{-3} a_{-4} \ldots, a_{-1}, a_1) \rightarrow (0.a_2 a_3 a_4 \ldots, 0.a_{-2} a_{-3} a_{-4} \ldots, a'_0, a_{-1}, a'_1) . \)

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

\( (0.a_2 a_3 a_4 \ldots, 0.a_{-2} a_{-3} a_{-4} \ldots, a'_0, a_{-1}, a'_1) \rightarrow (0.a'_1 a_2 a_3 \ldots, 0.a_{-2} a_{-3} a_{-4} \ldots, a_{-2}, a'_0, a_{-1}) . \)

Hence

\[ (x_b, y_b, p_b, q_b, s_b) = (0.a'_1 a_2 a_3 \ldots, 0.a_{-2} a_{-3} a_{-4} \ldots, a_{-2}, a'_0, a_{-1}) . \]

So we have:

\[ x_b = \frac{G(a_{i_DOD}) n^2 - G(a_{i_DOD}) n}{n} + \frac{x_a}{n} = 0.a'_1 + 0.0a_2 a_3 a_4 \ldots = 0.a'_1 a_2 a_3 \ldots \]

for \( G(a_{i_DOD}) n^2 - G(a_{i_DOD}) n = a'_1 a_{-1} a_1 - a'_0 a_{-1} 0 = a'_1 , \)

\[ y_b = y_a n - \left\lfloor y_a n \right\rfloor = a_{-2} a_{-3} a_{-4} \ldots - a_{-2} = 0.a_{-2} a_{-3} a_{-4} \ldots , \]

\[ p_b = \left\lfloor y_a n \right\rfloor = a_{-2} , \]

\[ q_b = G(a_{i_DOD}) = a'_0 , \]

\[ s_b = p_a = a_{-1} . \]

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 \( f_{GS} : R^5 \rightarrow R^5 \) belonging to \( H_1 \), that simulates this Turing machine.

**Proof.** Let us observe that \( a_{i_DOD} = p_a + \frac{q_a}{n} + \frac{s_a}{n^2} \) is in \( H_0 \).

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

To define the function \( f_{GS} : R^5 \rightarrow R^5 \) three different variables \( p_a \), \( q_a \) and \( s_a \) (each of these symbols is one of the digits occurring in \( a_{i_DOD} \)) have been used.
This allows to define $a_{POD}$ without a function $\lfloor x \rfloor$. This fact makes it possible to decrease the class of functions $f_{GS}$ in $\eta$-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 $\eta$-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