OBSERVABILITY OF TURING MACHINES: A REFINEMENT OF THE THEORY OF COMPUTATION

Yaroslav D. Sergeyev* **and Alfredo Garro**†‡

Dipartimento di Elettronica, Informatica e Sistemistica,
Università della Calabria,
87030 Rende (CS) – Italy

Abstract

The Turing machine is one of the simple abstract computational devices that can be used to investigate the limits of computability. In this paper, they are considered from several points of view that emphasize the importance and the relativity of mathematical languages used to describe the Turing machines. A deep investigation is performed on the interrelations between mechanical computations and their mathematical descriptions emerging when a human (the researcher) starts to describe a Turing machine (the object of the study) by different mathematical languages (the instruments of investigation). Together with traditional mathematical languages using such concepts as 'enumerable sets' and 'continuum' a new computational methodology allowing one to measure the number of elements of different infinite sets is used in this paper. It is shown how mathematical languages used to describe the machines limit our possibilities to observe them. In particular, notions of observable deterministic and non-deterministic Turing machines are introduced and conditions ensuring that the latter can be simulated by the former are established.

Key Words: Theory of automatic computations, observability of Turing machines, relativity of mathematical languages, infinite sets, Sapir-Whorf thesis.

*Yaroslav D. Sergeyev, Ph.D., D.Sc., holds a Full Professorship reserved for distinguished scientists at the University of Calabria, Rende, Italy. He is also Full Professor (a part-time contract) at the N.I. Lobatchevsky State University, Nizhni Novgorod, Russia and Affiliated Researcher at the Institute of High Performance Computing and Networking of the National Research Council of Italy. yaro@si.deis.unical.it

†Alfredo Garro, Ph.D., is Assistant Professor at the University of Calabria, Rende, Italy. garro@si.deis.unical.it

‡The authors thank the anonymous reviewers for their useful suggestions. This research was partially supported by the Russian Federal Program "Scientists and Educators in Russia of Innovations", contract number 02.740.11.5018.

1 Introduction

The fundamental nature of the concept *automatic computations* attracted a great attention of mathematicians (and later of computer scientists) since 1930's (see [5, 13, 14, 15, 17, 18, 21, 32] and more recent monographs [2, 6, 7, 11]). At that time, this strong impetus for understanding what is computable was actively supported by David Hilbert who believed that all of Mathematics could be precisely axiomatized. Several mathematicians from around the world proposed their independent definitions of what it means to be computable and what it means an automatic computing machine. In order to perform a rigorous study of sequential computations, they worked with different mathematical models of computing machines. Surprisingly, it has been discovered (see detailed discussions on this topic in, e.g., [2, 6, 7]) that all of these models were equivalent, e.g., anything computable in the λ-calculus is computable by a Turing machine.

In spite of the fact that the famous results of Church, Gödel, and Turing have shown that Hilbert's programme cannot be realized, the idea of finding an adequate set of axioms for one or another field of Mathematics continues to be among the most attractive goals for contemporary mathematicians as well. Usually, when it is necessary to define a concept or an object, logicians try to introduce a number of axioms describing the object. However, this way is fraught with danger because of the following reasons.

First, when we describe a mathematical object or concept we are limited by the expressive capacity of the language we use to make this description. A richer language allows us to say more about the object and a weaker language – less. Thus, development of the mathematical (and not only mathematical) languages leads to a continuous necessity of a transcription and specification of axiomatic systems. Second, there is no guarantee that the chosen axiomatic system defines 'sufficiently well' the required concept and a continuous comparison with practice is required in order to check the goodness of the accepted set of axioms. However, there cannot be again any guarantee that the new version will be the last and definitive one. Finally, the third limitation already mentioned above has been discovered by Gödel in his two famous incompleteness theorems (see [8]).

In linguistics, the relativity of the language with respect to the world around us has been formulated in the form of the Sapir-Whorf thesis (see [4, 10, 16, 23]) also known as the 'linguistic relativity thesis' (that has also interesting relations to the ideas of K.E. Iverson exposed in his Turing lecture [12]). As becomes clear from its name, the thesis does not accept the idea of the universality of language and postulates that the nature of a particular language influences the thought of its speakers. The thesis challenges the possibility of perfectly representing the world with language, because it implies that the mechanisms of any language condition the thoughts of its speakers.

In this paper, we study the relativity of mathematical languages in situations where they are used to observe and to describe automatic computations (we consider the traditional computational paradigm mainly following results of Turing

(see [32]) whereas emerging computational paradigms (see, e.g. [1, 19, 34, 33]) are not considered here). Let us illustrate the concept of the relativity of mathematical languages by the following example. In his study published in *Science* (see [9]), Peter Gordon describes a primitive tribe living in Amazonia – Pirahã – that uses a very simple numeral system[1] for counting: one, two, 'many'. For Pirahã, all quantities larger than two are just 'many' and such operations as 2+2 and 2+1 give the same result, i.e., 'many'. By using their weak numeral system Pirahã are not able to see, for instance, numbers 3, 4, and 5, to execute arithmetical operations with them, and, in general, to say anything about these numbers because in their language there are neither words nor concepts for that.

The numeral system of Pirahã has another interesting feature particularly interesting in the context of the study presented in this paper:

$$\text{'many'} + 1 = \text{'many'}, \quad \text{'many'} + 2 = \text{'many'}, \quad \text{'many'} + \text{'many'} = \text{'many'}. \quad (1)$$

These relations are very familiar to us in the context of our views on infinity used in the calculus

$$\infty + 1 = \infty, \qquad \infty + 2 = \infty, \qquad \infty + \infty = \infty. \quad (2)$$

Thus, the modern mathematical numeral systems allow us to distinguish a larger quantity of finite numbers with respect to Pirahã but give similar results when we speak about infinite numbers. Formulae (1) and (2) lead us to the following observation: *Probably our difficulty in working with infinity is not connected to the nature of infinity but is a result of inadequate numeral systems used to express infinite numbers.* Analogously, Pirahã do not distinguish numbers 3 and 4 not due to the nature of these numbers but due to the weakness of their numeral system.

This remark is important with respect to the computability context because of the following reason. Investigations of traditional computational models (we do not discuss emerging computational paradigms, see, e.g. [3]) executed so far used for studying infinite computational processes mathematical instruments developed by Georg Cantor (see [3]) who has shown that there exist infinite sets having different number of elements. In the theory of computations, two infinite sets – countable sets and continuum – are used mainly. Cantor has proved, by using his famous diagonal argument, that the cardinality, $\aleph_0$, of the set, $\mathbb{N}$, of natural numbers is less than the cardinality, C, of real numbers $x \in [0,1]$.

Cantor has also developed an arithmetic for the infinite cardinal numbers. Some of the operations of this arithmetic including $\aleph_0$ and C are given below:

$$\aleph_0 + 1 = \aleph_0, \qquad \aleph_0 + 2 = \aleph_0, \qquad \aleph_0 + \aleph_0 = \aleph_0,$$

[1] We remind that *numeral* is a symbol or group of symbols that represents a *number*. The difference between numerals and numbers is the same as the difference between words and the things they refer to. A *number* is a concept that a *numeral* expresses. The same number can be represented by different numerals. For example, the symbols '9', 'nine', and 'IX' are different numerals, but they all represent the same number.

$$C+1 = C, \qquad C+2 = C, \qquad C+C = C, \qquad C+\aleph_0 = C.$$

Again, it is possible to see a clear similarity with the arithmetic operations used in the numeral system of Pirahã.

Advanced contemporary numeral systems enable us to distinguish within 'many' various large finite numbers. As a result, we can use large finite numbers in computations and construct mathematical models involving them. Analogously, if we were be able to distinguish more infinite numbers probably we could understand better the nature of the sequential automatic computations (remind the famous phrase of Ludwig Wittgenstein: 'The limits of my language are the limits of my world.').

The goal of this paper is to study Turing machines using a new approach introduced in [24, 25, 26] and allowing one to write down different finite, infinite, and infinitesimal numbers by a finite number of symbols as particular cases of a unique framework. Its applications in several fields can be found in [24, 28, 29, 30, 31]. It is worthy to mention also that the new computational methodology has given a possibility to introduce the Infinity Computer (see [25] and the European patent [27]) working numerically with finite, infinite, and infinitesimal numbers (its software simulator has already been realized).

The rest of the paper is structured as follows. In Section 2, a brief introduction to the new methodology is given. Due to a rather unconventional character of the new methodology, the authors kindly recommend the reader to study the survey [26] (downloadable from [25]) before approaching Sections 3 – 5.

Section 3 presents some preliminary results regarding description of infinite sequences by using a new numeral system. Section 4 shows that the introduced methodology applied together with a new numeral system allows one to have a fresh look at mathematical descriptions of Turing machines. A deep investigation is performed on the interrelations between mechanical computations and their mathematical descriptions emerging when a human (the researcher) starts to describe a Turing machine (the object of the study) by different mathematical languages (the instruments of investigation). Mathematical descriptions of automatic computations obtained by using the traditional language and the new one are compared and discussed. An example of the comparative usage of both languages is given in Section 5 where they are applied for descriptions of deterministic and non-deterministic Turing machines. After all, Section 6 concludes the paper.

2 Methodology and a new numeral system

In this section, we give just a brief introduction to the methodology of the new approach [24, 26] dwelling only on the issues directly related to the subject of the paper. This methodology will be used in the subsequent sections to study Turing machines and to obtain some more detailed results related to the further understanding of what is effectively computable – the problem that was stated and widely discussed in [5, 32].

We start by introducing three postulates that will fix our methodological positions (having a strong applied character) with respect to infinite and infinitesimal quantities and Mathematics, in general.

Postulate 1. *There exist infinite and infinitesimal objects but human beings and machines are able to execute only a finite number of operations.*

Postulate 2. *We shall not tell* ***what are*** *the mathematical objects we deal with; we just shall construct more powerful tools that will allow us to improve our capabilities to observe and to describe properties of mathematical objects.*

Postulate 3. *The principle 'The part is less than the whole' is applied to all numbers (finite, infinite, and infinitesimal) and to all sets and processes (finite and infinite).*

In Physics, researchers use tools to describe the object of their study and the used instrument influences results of observations and restricts possibilities of observation of the object. Thus, there exists the philosophical triad – researcher, object of investigation, and tools used to observe the object. Postulates 1–3 emphasize existence of this triad in Mathematics and Computer Science, as well. Mathematical languages (in particular, numeral systems) are among the tools used by mathematicians to observe and to describe mathematical objects. As a consequence, very often difficulties that we find solving mathematical problems are related not to their nature but to inadequate mathematical languages used to solve them.

It is necessary to notice that due to the declared applied statement fixed by Postulates 1–3, such concepts as bijection, numerable and continuum sets, cardinal and ordinal numbers cannot be used in this paper because they belong to the theories working with different assumptions. As a consequence, the new approach is different also with respect to the non-standard analysis introduced in [22] and built using Cantor's ideas. However, the approach used here does not contradict Cantor. In contrast, it evolves his deep ideas regarding existence of different infinite numbers in a more applied way and can be viewed as a more strong lens of our mathematical microscope that allows one, e.g., not only to separate different classes of infinite sets but also to measure the number of elements of some infinite sets.

By accepting Postulate 1 we admit that it is not possible to have a complete description of infinite processes and sets due to our finite capabilities. For instance, we accept that we are not able to observe all elements of an infinite set (this issue will be discussed in detail hereinafter).

It is important to emphasize that our point of view on axiomatic systems is also more applied than the traditional one. Due to Postulate 2, mathematical objects are not defined by axiomatic systems that just determine formal rules for operating with certain numerals reflecting some properties of the studied mathematical objects.

Due to Postulate 3, infinite and infinitesimal numbers should be managed in the same manner as we are used to deal with finite ones. This Postulate in our opinion very well reflects organization of the world around us but in many traditional infinity theories it is true only for finite numbers. Due to Postulate 3, the traditional

point of view on infinity accepting such results as $\infty+1=\infty$ should be substituted in a way ensuring that $\infty+1>\infty$.

This methodological program has been realized in [24, 26] where a new powerful numeral system has been developed. This system gives a possibility to execute *numerical* computations not only with finite numbers but also with infinite and infinitesimal ones in accordance with Postulates 1–3. The main idea consists of measuring infinite and infinitesimal quantities by different (infinite, finite, and infinitesimal) units of measure.

A new infinite unit of measure has been introduced for this purpose in [24, 26] in accordance with Postulates 1–3 as the number of elements of the set $\mathbb{N}$ of natural numbers. It is expressed by a new numeral ① called *grossone*.

It is necessary to emphasize immediately that the infinite number ① is not either Cantor's $\aleph_0$ or ω. Particularly, it has both cardinal and ordinal properties as usual finite natural numbers. Formally, grossone is introduced as a new number by describing its properties postulated by the *Infinite Unit Axiom* (see [24, 26]). This axiom is added to axioms for real numbers similarly to addition of the axiom determining zero to axioms of natural numbers when integer numbers are introduced. Again, we speak about axioms of real numbers in sense of Postulate 2, i.e., axioms define formal rules of operations with numerals in a given numeral system.

Inasmuch as it has been postulated that grossone is a number, all other axioms for numbers hold for it, too. Particularly, associative and commutative properties of multiplication and addition, distributive property of multiplication over addition, existence of inverse elements with respect to addition and multiplication hold for grossone as for finite numbers. This means, for example, that the following relations hold for grossone, as for any other number

$$0\cdot①=①\cdot 0=0,\quad ①-①=0,\quad \frac{①}{①}=1,\quad ①^0=1,\quad 1^①=1,\quad 0^①=0. \qquad (3)$$

Let us comment upon the nature of grossone by some illustrative examples (see the survey [26] for a detailed discussion).

Example 2.1. Infinite numbers constructed using grossone can be interpreted in terms of the number of elements of infinite sets. For example, $①-2$ is the number of elements of a set $B=\mathbb{N}\backslash\{b_1,b_2\}$ where $b_1,b_2\in\mathbb{N}$. Analogously, $①+1$ is the number of elements of a set $A=\mathbb{N}\cup\{a\}$, where $a\notin\mathbb{N}$. Due to Postulate 3, integer positive numbers that are larger than grossone do not belong to $\mathbb{N}$ but also can be easily interpreted. For instance, $①^3$ is the number of elements of the set V, where

$$V=\{(a_1,a_2,a_3): a_1\in\mathbb{N},a_2\in\mathbb{N},a_3\in\mathbb{N}\}. \qquad \square$$

Example 2.2. Grossone has been introduced as the quantity of natural numbers. Similarly to the set

$$A=\{1,2,3,4,5\} \qquad (4)$$

having 5 elements where 5 is the largest number in A, ① is the largest *infinite* natural number[2] and ① $\in \mathbb{N}$. As a consequence, the set, $\mathbb{N}$, of natural numbers can be written in the form

$$\mathbb{N} = \{1, 2, 3, \quad \ldots \quad ① - 2, \ ① - 1, \ ①\}. \tag{5}$$

Traditional numeral systems did not allow us to see infinite natural numbers $\ldots ① - 2, ① - 1, ①$. Similarly, the Pirahã are not able to see finite natural numbers greater than 2. In spite of this fact, these numbers (e.g., 3 and 4) belong to $\mathbb{N}$ and are visible if one uses a more powerful numeral system. Thus, we have the same object of observation – the set $\mathbb{N}$ – that can be observed by different instruments – numeral systems – with different accuracies (see Postulate 2). □

As it has been mentioned above, the introduction of the numeral ① allows us to introduce various numerals that can be used to express integer positive numbers larger than grossone such as $①^2$, $①^3 - 4$, and also $2^①, 10^① + 3$, etc. (their meaning will be explained soon). This leads us to the necessity to introduce the set of *extended natural numbers* (including $\mathbb{N}$ as a proper subset) indicated as $\widehat{\mathbb{N}}$ where

$$\widehat{\mathbb{N}} = \{1, 2, \ldots, ① - 1, ①, ① + 1, ① + 2, ① + 3, \ldots, ①^2 - 1, ①^2, ①^2 + 1, \ldots\}. \tag{6}$$

It is useful to notice that, due to Postulates 1 and 2, the new numeral system cannot give answers to *all* questions regarding infinite sets. A mathematical language can allow one to formulate a question but not its answer. For instance, it is possible to formulate the question: 'What is the number of elements of the set $\widehat{\mathbb{N}}$?' but the answer to this question cannot be expressed within a numeral system using only ①. It is necessary to introduce in a reasonable way a more powerful numeral system by defining new numerals (for instance, ②, ③, etc.).

Example 2.3. Let us consider the set of even numbers, $\mathbb{E}$, from the traditional point of view. Cantor's approach establishes the following one-to-one correspondence with the set of all natural numbers, $\mathbb{N}$, in spite of the fact that $\mathbb{E}$ is a part of $\mathbb{N}$:

$$\begin{array}{lccccccc} \text{even numbers:} & 2, & 4, & 6, & 8, & 10, & 12, & \ldots \\ & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \\ \text{natural numbers:} & 1, & 2, & 3, & 4 & 5, & 6, & \ldots \end{array} \tag{7}$$

This result can be viewed in the following way: traditional mathematical tools do not allow us to distinguish inside the class of enumerable sets infinite sets having different number of elements.

From the new point of view, the one-to-one correspondence cannot be used as a *tool* for working with infinite sets because, due to Postulate 1, we are able to execute only a finite number of operations and the sets $\mathbb{E}$ and $\mathbb{N}$ are infinite.

[2] This fact is one of the important methodological differences with respect to non-standard analysis theories where it is supposed that infinite numbers do not belong to $\mathbb{N}$.

However, analogously to (5), the set, $\mathbb{E}$, of even natural numbers can be written (see [26] for a detailed discussion) in the form

$$\mathbb{E} = \{2, 4, 6 \quad \ldots \quad ① - 4,\ ① - 2,\ ①\}, \tag{8}$$

since ① is even and the number of elements of the set of even natural numbers is equal to $\frac{①}{2}$. Note that the next even number is $① + 2$ but it is not natural because $① + 2 > ①$ (see (6)), it is extended natural. Thus, we can write down not only initial (as it is done traditionally) but also the final part of (7) as follows

$$\begin{array}{cccccccccc} 2, & 4, & 6, & 8, & 10, & 12, & \ldots & ① - 4, & ① - 2, & ① \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & & \updownarrow & \updownarrow & \updownarrow \\ 1, & 2, & 3, & 4 & 5, & 6, & \ldots & \frac{①}{2} - 2, & \frac{①}{2} - 1, & \frac{①}{2} \end{array} \tag{9}$$

concluding so (7) in a complete accordance with Postulate 3. □

Note that record (9) does not affirm that we have established the one-to-one correspondence among *all* even numbers and a half of natural ones. We cannot do this because, due to Postulate 1, we can execute only a finite number of operations and the considered sets are infinite. The symbols '...' in (9) indicate that there are infinitely many numbers between 12 and $① - 4$ in the first line and between 6 and $\frac{①}{2} - 2$ in the second line. The record (9) affirms that for any even natural number expressible in the chosen numeral system it is possible to indicate the corresponding natural number in the lower row of (9) if it is also expressible in this numeral system.

We conclude the discussion upon Example 2.3 by the following remark. With respect to our methodology, the mathematical results obtained by Cantor in (7) and our results (9) do not contradict each other. *They both are correct with respect to mathematical languages used to express them.* This relativity is very important and it has been emphasized in Postulate 2. The result (7) is correct in Cantor's language and the more powerful language developed in [24, 26] allows us to obtain a more precise result (9) that is correct in the new language.

The choice of the mathematical language depends on the practical problem that is to be solved and on the accuracy required for such a solution. The result (7) just means that Cantor's mathematical tools do not allow one to distinguish two observed mathematical objects, namely, the number of elements of the sets $\mathbb{E}$ and $\mathbb{N}$ from the point of view of the number of their elements. If one is satisfied with this accuracy, this answer can be used (and *was used* since Cantor has published his results in 1870's) in practice.

However, if one needs a more precise result, it is necessary to introduce a more powerful mathematical language (a numeral system in this case) allowing one to express the required answer in a more accurate way. Obviously, it is not possible to mix languages. For instance, the question 'what is the result of the operation 'many'+4?', where 'many' belongs to the numeral system of Pirahã, is nonsense.

3 Infinite sequences

In the traditional definition of the Turing machine the notion of infinity is used in a strong form (see [32] and, e.g., [7]). First, the Turing machine has an infinite one-dimensional tape divided into cells and its outputs are computable (infinite) sequences of numerals. Second, an infinite sequence of operations can be executed by the machine and it is supposed the availability of an infinite time for the computation. Turing writes in pages 232 and 233 of [32]:

> ***Computing machines.***
> If an *a*-machine prints two kinds of symbols, of which the first kind (called figures) consists entirely of 0 and 1 (the others being called symbols of the second kind), then the machine will be called a computing machine. If the machine is supplied with a blank tape and set in motion, starting from the correct initial *m*-configuration, the subsequence of the symbols printed by it which are of the first kind will be called the *sequence computed by the machine*. The real number whose expression as a binary decimal is obtained by prefacing this sequence by a decimal point is called the *number computed by the machine*. [...]
>
> ***Circular and circle-free machines.***
> If a computing machine never writes down more than a finite number of symbols of the first kind it will be called *circular*. Otherwise it is said to be *circle-free*. [...]
>
> ***Computable sequences and numbers.***
> A sequence is said to be computable if it can be computed by a circle-free machine.

It is clear that the notion of the infinite sequence becomes very important for our study of the Turing machine. Thus, before considering the notion of the Turing machine from the point of view of the new methodology, let us explain how the notion of the infinite sequence can be viewed from the new positions.

Traditionally, an *infinite sequence* $\{a_n\}, a_n \in A$, $n \in \mathbb{N}$, is defined as a function having the set of natural numbers, $\mathbb{N}$, as the domain and a set A as the codomain. A *subsequence* $\{b_n\}$ is defined as a sequence $\{a_n\}$ from which some of its elements have been removed. In spite of the fact of the removal of the elements from $\{a_n\}$, the traditional approach does not allow one to register, in the case where the obtained subsequence $\{b_n\}$ is infinite, the fact that $\{b_n\}$ has less elements than the original infinite sequence $\{a_n\}$.

From the point of view of the new methodology, an infinite sequence can be considered in a dual way: either as an object of a mathematical study or as a mathematical instrument developed by human beings to observe other objects and processes (see Postulate 2). First, let us consider it as a mathematical object and show that the definition of infinite sequences should be done more precise within

the new methodology. The following result (see [24, 26]) holds. We reproduce here its proof for the sake of completeness.

Theorem 3.1. *The number of elements of any infinite sequence is less or equal to* ①.

Proof. The new numeral system allows us to express the number of elements of the set $\mathbb{N}$ as ①. Thus, due to the sequence definition given above, any sequence having $\mathbb{N}$ as the domain has ① elements.

The notion of subsequence is introduced as a sequence from which some of its elements have been removed. Due to Postulate 3, this means that the resulting subsequence will have less elements than the original sequence. Thus, we obtain infinite sequences having the number of members less than grossone. □

It becomes appropriate now to define the *complete sequence* as an infinite sequence containing ① elements. For example, the sequence of natural numbers is complete, the sequences of even and odd natural numbers are not complete because they have $\frac{①}{2}$ elements each (see [24, 26]). Thus, the new approach imposes a more precise description of infinite sequences than the traditional one.

To define a sequence $\{a_n\}$ in the new language, it is not sufficient just to give a formula for a_n, we should determine (as it happens for sequences having a finite number of elements) its number of elements and/or the first and the last elements of the sequence. If the number of the first element is equal to one, we can use the record $\{a_n : k\}$ where a_n is, as usual, the general element of the sequence and k is the number (that can be finite or infinite) of members of the sequence.

In connection with this definition the following question arises inevitably. Suppose that we have two sequences, for example, $\{b_n : k_1\}$ and $\{c_n : k_2\}$, where both k_1 and k_2 are infinite numbers such that $k_1 < ①$ and $k_2 < ①$ but $k_1 + k_2 > ①$. Can we create a new sequence, $\{d_n : k\}$, composed from both of them, for instance, as it is shown below

$$b_1,\ b_2,\ \ldots\ b_{k_1-2},\ b_{k_1-1},\ b_{k_1},\ c_1,\ c_2,\ \ldots\ c_{k_2-2},\ c_{k_2-1},\ c_{k_2}$$

and which will be the value of the number of its elements k?

The answer to this question is 'no' because due to Theorem 3.1, a sequence cannot have more than ① elements. Thus, the longest sequence is $\{d_n : ①\}$. After arriving to the last element $d_①$, the sequence $\{d_n : ①\}$ will stop. However, the second sequence can then be started.

Example 3.1. Suppose that $k_1 = \frac{2①}{5}$ and $k_2 = \frac{4①}{5}$. Then starting from the element b_1 we can arrive at maximum to the element $c_{\frac{3①}{5}}$ being the element $d_①$ in the sequence $\{d_n : ①\}$ which we construct. Therefore, $k = ①$ and

$$\underbrace{b_1,\ \ldots\ b_{\frac{2①}{5}},\ c_1,\ \ldots c_{\frac{3①}{5}}}_{① \text{ elements}},\ \underbrace{c_{\frac{3①}{5}+1},\ldots\ c_{\frac{4①}{5}}}_{\frac{①}{5} \text{ elements}}.$$

The remaining members $c_{\frac{3①}{5}+1}, \ldots\, c_{\frac{4①}{5}}$ of the sequence $\{c_n : \frac{4①}{5}\}$ will form the second sequence, $\{g_n : l\}$ having $l = \frac{4①}{5} - \frac{3①}{5} = \frac{①}{5}$ elements. Thus, we have formed two sequences, the first of them is complete and the second is not. □

We have already seen the influence of Postulates 2 and 3 on the notion of the infinite sequence. Let us study now what Postulate 1 gives us in this context. First, since the object of the study – the sequence – has an infinite number of members, it follows from Postulate 1 that we cannot observe all of them. We can observe only a finite number of its elements, precisely, those members of the sequence for which there exist the corresponding numerals in the chosen numeral system.

Example 3.2. Let us consider the numeral system, $\mathcal{P}$, of Pirahã able to express only numbers 1 and 2. If we add to $\mathcal{P}$ the new numeral ①, we obtain a new numeral system (we call it $\widehat{\mathcal{P}}$). Let us consider now a sequence of natural numbers $\{n : ①\}$. It goes from 1 to ① (note that both numbers, 1 and ①, can be expressed by numerals from $\widehat{\mathcal{P}}$). However, the numeral system $\widehat{\mathcal{P}}$ is very weak and it allows us to observe only ten numbers from the sequence $\{n : ①\}$ represented by the following numerals

$$\underbrace{1, 2}_{finite}, \quad \ldots \quad \underbrace{\frac{①}{2} - 2, \frac{①}{2} - 1, \frac{①}{2}, \frac{①}{2} + 1, \frac{①}{2} + 2}_{infinite}, \quad \ldots \quad \underbrace{① - 2, ① - 1, ①}_{infinite}. \quad (10)$$

The first two numerals in (10) represent finite numbers, the remaining eight numerals express infinite numbers, and dots represent members of the sequence of natural numbers that are not expressible in $\widehat{\mathcal{P}}$ and, therefore, cannot be observed if one uses only this numeral system for this purpose. □

Note that Pirahã are not able to see finite numbers larger than 2 using their weak numeral system but these numbers are visible if one uses a more powerful numeral system. In particular, this means that when we speak about sets (finite or infinite) it is necessary to take care about tools used to describe a set. In order to introduce a set, it is necessary to have a language (e.g., a numeral system) allowing us to describe both the form of its elements in a way and the number of its elements. For instance, the set A from (4) cannot be defined using the mathematical language of Pirahã.

Analogously, the words 'the set of all finite numbers' do not define a set from our point of view. It is always necessary to specify which instruments are used to describe (and to observe) the required set and, as a consequence, to speak about 'the set of all finite numbers expressible in a fixed numeral system'. For instance, for Pirahã 'the set of all finite numbers' is the set $\{1, 2\}$ and for another Amazonian tribe – Mundurukú[3] – 'the set of all finite numbers' is the set A from (4). As it happens in Physics, the instrument used for an observation bounds the possibility

[3]Mundurukú (see [20]) fail in exact arithmetic with numbers larger than 5 but are able to compare and add large approximate numbers that are far beyond their naming range. Particularly, they use the words 'some, not many' and 'many, really many' to distinguish two types of large numbers (in this connection think about Cantor's $\aleph_0$ and $\aleph_1$).

of the observation and determines its accuracy. It is not possible to say what we shall see during our observation if we have not clarified which instruments will be used to execute the observation.

Let us consider now infinite sequences as one of the instruments used by mathematicians to study the world around us and other mathematical objects and processes. The first immediate consequence of Theorem 3.1 is that any *sequential* process can have at maximum ① elements. This means that a process of sequential observations of any object cannot contain more than ① steps[4]. Due to Postulate 1, we are not able to execute any infinite process physically but we assume the existence of such a process. Moreover, again due to Postulate 1, only a finite number of observations of elements of the considered infinite sequence can be executed by a human who is limited by the numeral system used for observation. However, the researcher can choose how to organize the required sequence of observations and which numeral system to use for it, defining so which elements of the object he/she can observe. This situation is exactly the same as in natural sciences: before starting to study a physical object, a scientist chooses an instrument and its accuracy for the study.

Example 3.3. Let us consider the set, $\widehat{\mathbb{N}}$, of extended natural numbers from (6) as an object of our observation. Suppose that we want to organize the process of the sequential counting of its elements. Then, due to Theorem 3.1, starting from the number 1 this process can arrive at maximum to ①. If we consider the complete counting sequence $\{n : ①\}$, then we obtain

$$\underbrace{1,2,\ 3,\ 4,\ \ldots\ ①^{-}2, ①^{-}1, ①}_{① \text{ steps}}, ①^{+}1, ①^{+}2, ①^{+}3, \ldots \tag{11}$$

In this formula, a more powerful (with respect to $\widehat{\mathcal{P}}$ from (10)) numeral system, $\widetilde{\mathcal{P}}$, is used. It allows us to see also numbers three and four through the numerals 3 and 4 and, of course, such numbers as $① - 4, \frac{①}{3}, \frac{①}{4} - 3$, and other numbers that can be viewed through numerals obtained as combinations of symbols '+', '-', and '/' and numerals 1, 2, 3, 4, and ① similarly to (10) (we assume that finite numbers larger than 4 are not expressible in $\widetilde{\mathcal{P}}$). We omit them in the record (11) due to a straightforward similarity with (10).

Analogously, if we start the process of the sequential counting from 3, the

[4]It is worthy to notice a deep relation of this observation to the Axiom of Choice. Since Theorem 3.1 states that any sequence can have at maximum ① elements, so this fact holds for the process of a sequential choice, as well. As a consequence, it is not possible to choose sequentially more than ① elements from a set. This observation also emphasizes the fact that the parallel computational paradigm is significantly different with respect to the sequential one because p parallel processes can choose $p \cdot ①$ elements from a set.

process arrives at maximum to $①+2$:

$$\underbrace{1,2,3,4,\ \ldots\ ①^{-}2,①^{-}1,①,①^{+}1,①^{+}2}_{①\text{ steps}},①^{+}3,\ldots$$

The corresponding complete sequence used in this case is $\{n+2 : ①\}$. We can also change the length of the step in the counting sequence and consider, for instance, the complete sequence $\{2n-1 : ①\}$:

$$\underbrace{1,2,3,4,\ \ldots\ ①^{-}1,①,①^{+}1,①^{+}2,\ \ldots\ 2①^{-}3,2①^{-}2,2①^{-}1}_{①\text{ steps}},2①,2①^{+}1,\ldots$$

If we use again the numeral system $\widetilde{\mathcal{P}}$, then among finite numbers it allows us to see only numbers 1 and 3 because already the next number in the sequence, 5, is not expressible in $\widetilde{\mathcal{P}}$. The last two elements of the sequence are $2①-3$ and $2①-1$ and $\widetilde{\mathcal{P}}$ allows us to observe them. □

The introduced definition of a sequence allows us to work not only with the first but also with the last element of any sequence (if they are expressible in the chosen numeral system) independently whether it has a finite or an infinite number of elements. Let us use this new definition together with Postulate 2 for studying infinite sets of numerals, in particular, for calculating the number of points at the interval $[0,1)$ (see [24, 26]). To do this we need a definition of the term 'point' and mathematical tools to indicate a point. Since this concept is one of the most fundamental, it is very difficult to find an adequate definition. If we accept (as is usually done in modern Mathematics) that a *point A* belonging to the interval $[0,1)$ is determined by a numeral *x*, $x \in \mathbb{S}$, called *coordinate of the point A* where $\mathbb{S}$ is a set of numerals, then we can indicate the point *A* by its coordinate *x* and we are able to execute the required calculations.

It is worthwhile to emphasize that we have not postulated that *x* belongs to the set, $\mathbb{R}$, of real numbers as it is usually done, because we can express coordinates only by numerals and different choices of numeral systems lead to various sets of numerals. This situation is a direct consequence of Postulate 2 and is typical for natural sciences where it is well known that instruments influence the results of observations. Remind again the work with a microscope: we decide the level of the precision we need and obtain a result which is dependent on the chosen level of accuracy. If we need a more precise or a more rough answer, we change the lens of our microscope.

We should decide now which numerals we shall use to express coordinates of the points. After this choice we can calculate the number of numerals expressible in this system and, as a result, we obtain the number of points at the interval $[0,1)$. Different variants (see [24, 26]) can be chosen depending on the precision level we

want to obtain. For instance, we can choose a positional numeral system with a finite radix b that allows us to work with numerals

$$(0.a_{-1}a_{-2}\ldots a_{-(①-1)}a_{-①})_b, \quad a_{-i} \in \{0,1,\ldots b-2,b-1\}, \quad 1 \le i \le ①. \quad (12)$$

Then, the number of numerals (12) gives us the number of points within the interval $[0,1)$ expressed by these numerals. Note that a number using the positional numeral system (12) cannot have more than grossone digits (contrarily to sets discussed in Example 3.3) because a numeral having $g > ①$ digits would not be observable in a sequence. In this case such a record becomes useless in sequential computations because it does not allow one to identify numbers since $g - ①$ numerals remain non observed.

Theorem 3.2. *If coordinates of points $x \in [0,1)$ are expressed by numerals (12), then the number of the points x over $[0,1)$ is equal to $b^①$.*

Proof. In the numerals (12) there is a sequence of digits, $a_{-1}a_{-2}\ldots a_{-(①-1)}a_{-①}$, used to express the fractional part of the number. Due to the definition of the sequence and Theorem 3.1, any infinite sequence can have at maximum ① elements. As a result, there is ① positions on the right of the dot that can be filled in by one of the b digits from the alphabet $\{0,1,\ldots,b-1\}$. Thus, we have $b^①$ combinations to express the fractional part of the number. Hence, the positional numeral system using the numerals of the form (12) can express $b^①$ numbers. □

Corollary 3.1. *The number of numerals*

$$(a_1a_2a_3\ldots a_{①-2}a_{①-1}a_{①})_b, \quad a_i \in \{0,1,\ldots b-2,b-1\}, \quad 1 \le i \le ①, \quad (13)$$

expressing integer numbers in the positional system with a finite radix b in the alphabet $\{0,1,\ldots b-2,b-1\}$ is equal to $b^①$.

Proof. The proof is a straightforward consequence of Theorem 3.2 and is so omitted. □

Corollary 3.2. *If coordinates of points $x \in (0,1)$ are expressed by numerals (12), then the number of the points x over $(0,1)$ is equal to $b^① - 1$.*

Proof. The proof follows immediately from Theorem 3.2. □

Note that Corollary 3.2 shows that it becomes possible now to observe and to register the difference of the number of elements of two infinite sets (the interval $[0,1)$ and the interval $(0,1)$, respectively) even when only one element (the point 0) has been excluded from the first set in order to obtain the second one.

4 The Turing machines viewed through the lens of the new methodology

In the previous section, we studied static infinite mathematical objects – sets – by using infinite sequences as tools of the research. Let us establish now what can we

say with respect to physical and mathematical processes viewed as objects of observation having in mind the triad 'object, instrument, and researcher' emphasized by Postulate 2. Our main attention will be focused on processes related to the Turing machines and various manifestations of infinity taking place during the work of the machines and during mathematical descriptions of the machines performed by researchers.

Remind that traditionally, a Turing machine (see, e.g., [11, 32]) can be defined as a 7-tuple

$$\mathcal{M} = \left\langle Q, \Gamma, \overline{b}, \Sigma, q_0, F, \delta \right\rangle, \tag{14}$$

where Q is a finite and not empty set of states; Γ is a finite set of symbols; $\overline{b} \in \Gamma$ is a symbol called blank; $\Sigma \subseteq \{\Gamma - \overline{b}\}$ is the set of input/output symbols; $q_0 \in Q$ is the initial state; $F \subseteq Q$ is the set of final states; $\delta : \{Q - F\} \times \Gamma \mapsto Q \times \Gamma \times \{R, L, N\}$ is a partial function called the transition function, where L means left, R means right, and N means no move.

Specifically, the machine is supplied with: (i) a *tape* running through it which is divided into cells each capable of containing a symbol $\gamma \in \Gamma$, where Γ is called the tape alphabet, and $\overline{b} \in \Gamma$ is the only symbol allowed to occur on the tape infinitely often; (ii) an *head* that can read and write symbols on the tape and move the tape left and right one and only one cell at a time. The behavior of the machine is specified by its *transition function* δ and consists of a sequence of computational steps; in each step the machine reads the symbol under the head and applies the *transition function* that, given the current state of the machine and the symbol it is reading on the tape, specifies (if it is defined for these inputs): (i) the symbol $\gamma \in \Gamma$ to write on the cell of the tape under the head; (ii) the move of the tape (L for one cell left, R for one cell right, N for no move); (iii) the next state $q \in Q$ of the machine.

Following Turing (see [32]), we consider machines that have finite input and output alphabets, inputs of a finite length, a finite number of internal states but can work an infinite time and are able to produce outputs of an infinite length. Hereinafter such a machine is called an imaginary Turing machine, $\mathcal{T}^I$. In order to study the limitations of practical automatic computations, we also consider machines, $\mathcal{T}^{\mathcal{P}}$, that can be constructed physically. They are identical to $\mathcal{T}^I$ but are able to work only a finite time and can produce only finite outputs. We study both kinds of machines:

- from the point of view of their outputs called by Turing 'computable numbers' or 'computable sequences';
- from the point of view of algorithms that can be executed by a Turing machine.

4.1 Computable sequences

Let us consider first a physical machine $\mathcal{T}^{\mathcal{P}}$. We suppose that its output is written on the tape using an alphabet Σ containing symbols $\{0, 1, \ldots b-2, b-1\}$ where

b is a finite number (Turing in [32] uses $b = 10$). Thus, the output consists of a sequence of digits that can be viewed as a number in a positional system $\mathcal{B}$ with the radix b.

It follows from Postulate 1 (reflecting a fundamental law existing in the real world) that $\mathcal{T}^{\mathcal{P}}$ should stop after a finite number of iterations. The magnitude of this value depends on the physical construction of the machine, the way the notion 'iteration' has been defined, etc., but in any case this number is finite. The machine stops in two cases: (i) it has finished execution of its program and stops; (ii) it has not finished execution of the program and stops just because of a breakage of some of its components. In both cases the output sequence

$$(a_1a_2a_3\ldots a_{k-1},a_k)_b, \quad a_i \in \{0,1,\ldots b-2,b-1\}, \quad 1 \le i \le k, \tag{15}$$

of $\mathcal{T}^{\mathcal{P}}$ has a finite length k. Suppose that the maximal length of the output sequence that can be computed by $\mathcal{T}^{\mathcal{P}}$ is equal to a finite number $K_{\mathcal{T}^{\mathcal{P}}}$. Then it follows $k \le K_{\mathcal{T}^{\mathcal{P}}}$. This means that there exist problems that cannot be solved by $\mathcal{T}^{\mathcal{P}}$ if the length of the output necessary to write down the solution outnumbers $K_{\mathcal{T}^{\mathcal{P}}}$. If a machine $\mathcal{T}^{\mathcal{P}}$ has stopped to write the output after it has printed $K_{\mathcal{T}^{\mathcal{P}}}$ symbols then it is not clear whether the obtained output is a solution or just a result of the depletion of its computational resources. In particular, with respect to the halting problem it follows that all algorithms stop on $\mathcal{T}^{\mathcal{P}}$.

Let us call a person working with the machine and reading the output as a *researcher* (or a *user*). Then, in order to be able to read and to understand the output, the researcher should have his/her own positional numeral system $\mathcal{U}$ with an alphabet $\{0,1,\ldots u-2,u-1\}$ where $u \ge b$ from (15). Otherwise, the output cannot be understood and decoded by the user. Moreover, he/she should be able to read and to interpret output sequences of symbols with the length $K_{\mathcal{U}} \ge K_{\mathcal{T}^{\mathcal{P}}}$. If the situation $K_{\mathcal{U}} < K_{\mathcal{T}^{\mathcal{P}}}$ holds, then this means that the user is not able to interpret the obtained result. Thus, the number $K^* = \min\{K_{\mathcal{U}}, K_{\mathcal{T}^{\mathcal{P}}}\}$ defines the length of the outputs that can be computed and then observed and interpreted by the user. As a consequence, algorithms producing outputs having more than K^* positions become less interesting from the practical point of view.

It is possible to make analogous considerations with respect to alphabets and numeral systems used for input sequences restricting so again the number of algorithms useful from the practical point of view. Finally, the algorithm should be written down someway. This operation is executed by using an alphabet and a numeral system used for writing down the algorithm introduces limitations to the algorithms that can be proposed for executing them on a machine. These considerations are important because on the one hand, they establish limits of practical automatic computations and on the other hand, they emphasize the role of numeral systems in codifying algorithms and interpreting results of computations.

Let us turn now to imaginary Turing machines $\mathcal{T}^{I}$. Such a machine can produce outputs (15) with an infinite number of symbols k. In order to be *observable in a sequence*, an output should have $k \le ①$ (remind that the positional numeral

system $\mathcal{B}$ includes numerals being a *sequence* of digits, in order to be a numeral, the output should have $k \leq ①$). Outputs observable in a sequence play an important role in the further consideration.

Theorem 4.1. *Let M be the number of all possible complete computable sequences that can be produced by imaginary Turing machines using outputs (15) being numerals in the positional numeral system $\mathcal{B}$. Then it follows $M \leq b^{①}$.*

Proof. This result follows from the definitions of the complete sequence and the positional numeral system considered together with Theorem 3.2 and Corollary 3.1. □

Corollary 4.1. *Let us consider an imaginary Turing machine $\mathcal{T}^I$ working with the alphabet $\{0,1,2\}$ and computing the following complete computable sequence*

$$\underbrace{0,1,2,0,1,2,0,1,2,\ \ldots\ 0,1,2,0,1,2}_{① \textit{ positions}}. \tag{16}$$

Then imaginary Turing machines working with the output alphabet $\{0,1\}$ cannot produce observable in a sequence outputs that codify and compute (16).

Proof. Since the numeral 2 does not belong to the alphabet $\{0,1\}$ it should be coded by more than one symbol. One of the coding using the minimal number of symbols in the alphabet $\{0,1\}$ necessary to code numbers $0,1,2$ is $\{00,01,10\}$. Then the output corresponding to (16) and computed in this codification should be

$$00,01,10,00,01,10,00,01,10,\ \ldots\ 00,01,10,00,01,10. \tag{17}$$

Since the output (16) contains grossone positions, the output (17) would contain $2①$ positions. However, in order to be observable in a sequence, (16) should not have more than grossone positions. This fact completes the proof. □

At first glance results established by Theorem 4.1 and Corollary 4.1 sound quite unusual for a person who studied the behavior of Turing machines on infinite computable sequences using traditional mathematical tools. However, they do not contradict each other. Theorem 4.1 and Corollary 4.1 do not speak about *all* Turing machines. They consider only those machines that produce complete output sequences. If the object of observation (in this case – the output) contains more than grossone elements, it cannot be observed and, therefore, is less interesting from the point of view of practical computations.

It is important to emphasize that these results are in line with the situation that we have in the real world with a finite number of positions in the output sequences. For instance, suppose that a physical Turing machine $\mathcal{T}^{\mathcal{P}}$ has 6 positions at its output, the numeral system $\{0,1,2\}$, and the sequence $0,1,2,0,1,2$ is computed. Then there does not exist a Turing machine working with the output alphabet $\{0,1\}$ able to calculate the sequence $0,1,2,0,1,2$ using the output having 6 positions.

In order to understand Theorem 4.1 and Corollary 4.1 better, let us return to the Turing machine as it has been described in [32] and comment upon connections

between the traditional results and the new ones. First, it is necessary to mention that results of Turing and results of Theorems 3.2, 4.1, and Corollary 4.1 have been formulated using different mathematical languages. The one used by Turing has been developed by Cantor and did not allow Turing to distinguish within continuum various sets having different number of elements. The new numeral system using grossone allows us to do this.

Cantor has proved, by using his famous diagonal argument, that the number of elements of the set $\mathbb{N}$ is less than the number of real numbers at the interval $[0,1)$ *without calculating the latter*. To do this, he expressed real numbers in a positional numeral system. We have shown that this number will be different depending on the radix b used in the positional system to express real numbers. However, all of the obtained numbers, $b^{①}$, are larger than the number of elements of the set of natural numbers, ①.

Thus, results presented in Theorem 4.1 and Corollary 4.1 should be considered just as a more precise analysis of the situation related to the existence of different infinities discovered by Cantor. The usage of a more powerful numeral system gives a possibility to distinguish and to describe more mathematical objects within the continuum, in the same way as the usage of a stronger lens in a microscope gives a possibility to distinguish more objects within an object that seems to be indivisible when viewed by a weaker lens.

As a consequence, the mathematical results obtained by Turing and those presented in Theorems 3.2, 4.1, and Corollary 4.1 do not contradict each other. *They are correct with respect to mathematical languages used to express them and correspond to different accuracies of the observation.* Both mathematical languages observe and describe the same object – computable sequences – but with different accuracies. This fact is one of the manifestations of the relativity of mathematical results formulated by using different mathematical languages.

Another manifestation of this relativity is obviously related to the concept of the universal Turing machine and to the process of establishing equivalence between machines. Notice that Theorem 4.1 and Corollary 4.1 emphasize dependence of the outputs of Turing machines on a finite alphabet $\{0,1,\ldots b-2,b-1\}$ used for writing down computable sequences. Therefore, when a researcher describes a Turing machine, there exists the dependence of the description on the finite numeral system used by the researcher. First, the description is limited by alphabets $\{0,1,\ldots b-2,b-1\}$ known to the humanity at the present situation. Second, by the maximal length, $K_{\mathcal{U}}$, of the sequence of symbols written in the fixed alphabet that the researcher is able to read, to write, and to understand.

It is not possible to describe a Turing machine (the object of the study) without the usage of a numeral system (the instrument of the study). Our possibilities to observe and to describe Turing machines and to count their number are limited by the numeral systems known to the humanity at the moment. Again, as it happens in natural sciences, the tools used in the study limit the researcher. As a result, it becomes not possible to speak about an absolute number of *all possible Turing machines* $\mathcal{T}^I$. It is always necessary to speak about the number of all possible

Turing machines $\mathcal{T}^I$ expressible in a fixed numeral system (or in a group of them).

The same limitations play an important role in the process of simulating one machine $\mathcal{T}^I$ by another. In order to be able to execute this operation it is necessary to calculate the respective description number (see [32]) and this will be possible only for description numbers expressible in the finite alphabets known at the current moment to the researcher and the length of these numbers will be limited by the number $K_{\mathcal{U}}$. A machine $\mathcal{T}^I$ having the description number not satisfying these constraints cannot be simulated because the instrument – a numeral system – required for such re-writing is not powerful enough (as usual, devil is in the details).

Let us consider now from positions of the new numeral system including grossone the situation related to the enumerability of machines $\mathcal{T}^I$ studied by Turing.

Theorem 4.2. *The maximal number of complete computable sequences produced by imaginary Turing machines that can be enumerated in a sequence is equal to* ①.

Proof. This result follows from the definition of a complete sequence. □

Let us consider the results of Theorems 4.1 and 4.2 together. Theorem 4.1 gives an upper bound for the number of complete computable sequences that can be computed using a fixed radix b. However, we do not know how many of $b^{①}$ sequences can be results of computations of a Turing machine. Turing establishes that their number is enumerable. In order to obtain this result, he used the mathematical language developed by Cantor and this language did not allow him to distinguish sets having different infinite numbers of elements, e.g., in the traditional language that he used the sets of even, natural, and integer numbers all are enumerable.

The introduction of grossone gives a possibility to execute a more precise analysis and to determine that these sets have different numbers of elements: $\frac{①}{2}$, ①, and $2① + 1$, respectively. If the number of complete computable sequences, $M_{\mathcal{T}^I}$, is larger than grossone, then there can be different sequential enumerating processes that enumerate complete computable sequences in different ways. Theorem 4.2 states that, in any case, each of these enumerating sequential processes cannot contain more than grossone members.

We conclude this subsection by noticing that the results presented in it establish limitations for the number of computable sequences only from the point of view of the output sequences and their alphabets. An analogous analysis can be done using limitations imposed by the number of states of Turing machines, their inputs, and the respective finite alphabets, as well.

4.2 Processes of automatic computations and their descriptions

First, we take notice that if we want to observe a process of computations A performed by a Turing machine ($\mathcal{T}^{\mathcal{P}}$ or $\mathcal{T}^I$) while it executes an algorithm, then we do it by executing observations of the machine in a sequence of moments. In fact, it is not possible to organize a *continuous* observation of the machine. Any instrument used for an observation has its accuracy and there will always be a minimal period of time related to this instrument allowing one to distinguish two different

moments of time and, as a consequence, to observe (and to register) the states of the object in these two moments. In the period of time passing between these two moments the object remains unobservable.

Hence, the observations are made in a sequence (that is an instrument of the research) and the process of computations A is the object of the study. In the simplest case we observe A only two times: at the starting point when we supply the input data and at the ending point of the process of computation when we read the results. In alternative, observations are made to look at intermediate results or even at particular moves of the parts of the machine (e.g., reading a symbol, writing a symbol, etc.).

On the one hand, since our observations are made in a sequence, it follows from Theorem 3.1 that the process of observations can have at maximum ① elements. This means that inside a computational process it is possible to fix more than grossone steps (defined someway) but it is not possible to count them one by one in a sequence containing more than grossone elements. For instance, in a time interval $[0, 1)$, numerals (12) can be used to identify moments of time but not more than grossone of them can be observed in a sequence.

On the other hand, it is important to stress that any process itself, considered independently on the researcher, is not subdivided in iterations, intermediate results, moments of observations, etc. This is a direct consequence of Postulate 2, the consequence that is also in line with the Sapir–Whorf thesis, particularly, with results of Whorf (see [4]) related to his analysis of the differences between Western languages and the Hopi language (a Uto-Aztecan language spoken by the Hopi people of northeastern Arizona, USA). Analyzing the relationship between language, thought, and reality in these two types of languages (see also recent experimental data and the relative discussion in [10, 16]) Whorf raises a barrier between them.

Western languages tend to analyze reality as objects in space. There exist other languages, including many Native American languages, that are oriented towards processes. To monolingual speakers of such languages, the constructions of Western languages related to objects and separate events may make little sense. On the other hand, due to Whorf, the relativistic physics – a subject being very hard for understanding for a Western language speaker – a Hopi speaker would find fundamentally easier to grasp. Whorf writes:

> We dissect nature along lines laid down by our native language. The categories and types that we isolate from the world of phenomena we do not find there because they stare every observer in the face; on the contrary, the world is presented in a kaleidoscope flux of impressions which has to be organized by our minds – and this means largely by the linguistic systems of our minds. We cut nature up, organize it into concepts, and ascribe significances as we do, largely because we are parties to an agreement to organize it in this way – an agreement that holds throughout our speech community and is codified in the patterns of our language $[\ldots]$

The Sapir–Whorf thesis is interesting for us because, in a complete accordance with our methodological positions, it separates the object of observations from its representation by one or another language. In particular, with respect to automatic computations we emphasize that a machine (a physical or an imaginary one) executing a computation does not distinguish an importance of one moment during the execution of an action with respect to another and does not count them. Certain milestones inside a process (computational steps defined someway, operations, iterations, etc.) are introduced from outside of the studied process by the researcher because these specific points are interesting for the observer for some reasons and can be expressed in his language. The notion 'sequence' is a tool invented by human beings, it is a part of the modern mathematical languages (developed mainly in the frame of Western languages dissecting processes in separate events), it does not take part of the object of the study.

When we speak about a computer executing iterations of a certain algorithm, *we* subdivide the process of computations on iterations and *we* count them. As a result, it is necessary to speak about the computational power of computers (in particular, of Turing machines) coupled with our possibilities to use them, to follow computational processes, to be able to provide input data, and to read results of computations. The understanding of the fact that computations executed by a computer and our observations and descriptions of these computations are different processes lead to the necessity to rethink such notions as *iteration* and *algorithm*.

At the moment when we decide what is an iteration of our algorithm, we are choosing the instrument of our investigation and the further results will depend on the chosen accuracy (or granularity) of observations. For instance, with respect to Turing machines an iteration can be a single operation of the machine such as reading a symbol from the tape, or moving the tape, etc. Another possible example of such a choice (that is usually used in the Computer Science literature) is to observe the machine when its configuration has been changed. All these choices produce different sequences of observations that form an algorithm if we add to them an input and an output being the symbols present on the tape of $\mathcal{T}^I$ at the first and at the last observation, respectively.

In order to conclude our discussion on the notion of the algorithm it is necessary to remind that any sequence cannot contain more then grossone steps. Thus, after we have chosen what is the iteration of our algorithm, the maximal number of these iterations cannot outnumber ①. As usual, the choice of the numeral system used to describe iterations and their results determines what will be observable for the researcher. Similarly, the choice of the numeral system (and, in general, of the mathematical language) used to describe the algorithm will limit the type of the algorithms that can be described.

The notion of the result of a computation on $\mathcal{T}^I$ has the same sense as it was for $\mathcal{T}^{\mathcal{P}}$. If $\mathcal{T}^I$ has not stopped after ① observations, then this means that we have finished our possibilities of observations and we cannot say whether the symbols present at the tape during this observation are effectively the solution to the problem. By a complete analogy with $\mathcal{T}^{\mathcal{P}}$, computations finish either because the ma-

chine $\mathcal{T}^I$ stops or because we are not any more able to observe computations (since $\mathcal{T}^I$ is an imaginary one, the possibility of its breakage is not taken into consideration). In particular, this means that with respect to the halting problem all algorithms stop but this does not mean that the obtained result is a solution.

The analysis given above shows us that it is not possible to speak about the computational power of a Turing machine without taking into consideration a number of limitations introduced by the languages. Among them there are at least the following: the language used to describe the algorithm and iterations it consists of; the language used to describe the process of computations; and the language used to describe the Turing machine itself (its input and output alphabets, its states, etc.). Notice that this situation with a description of automatic computations is just a particular case of the situation emphasized by Postulate 2: when we study an object it is necessary to be aware of the accuracy and the capability of instruments used for the study.

The obtained picture of the computability is significantly richer and complex with respect to traditional views (see [5, 6, 7, 13, 14, 15, 17, 21, 32]). The classic Turing theory contains a number of theoretical results showing the same computational power of different variants of Turing machines and establishing that the differences among machines $\mathcal{T}_1^I$ and $\mathcal{T}_2^I$ result only in the different number of steps that will be necessary to each machine for computing the required output. Some of the limitations on this point of view have been already discussed in Section 4.1. In this section, we have emphasized a number of additional limitations. Again, as it was in Section 4.1, the difference with the traditional results is not a contradiction. These differences arise because the mathematical language used for these traditional studies did not allow people to see the differences among various models of computations.

5 Usage of traditional and new languages for comparing deterministic and non-deterministic Turing machines

In order to illustrate the new way of reasoning, let us discuss the traditional and new results regarding the computational power of deterministic and non-deterministic Turing machines. For simplicity, we do not take into consideration limitations described in Section 4.1. Let us first remind the traditional point of view.

A non-deterministic Turing machine (see [11]) can be defined (cf. (14)) as a 7-tuple

$$\mathcal{M}_N = \left\langle Q, \Gamma, \bar{b}, \Sigma, q_0, F, \delta_N \right\rangle, \tag{18}$$

where Q is a finite and not empty set of states; Γ is a finite set of symbols; $\bar{b} \in \Gamma$ is a symbol called blank; $\Sigma \subseteq \{\Gamma - \bar{b}\}$ is the set of input/output symbols; $q_0 \in Q$ is the initial state; $F \subseteq Q$ is the set of final states; $\delta_N : \{Q - F\} \times \Gamma \mapsto \mathcal{P}(Q \times \Gamma \times \{R, L, N\})$ is a partial function called the transition function, where L means left, R means right, and N means no move.

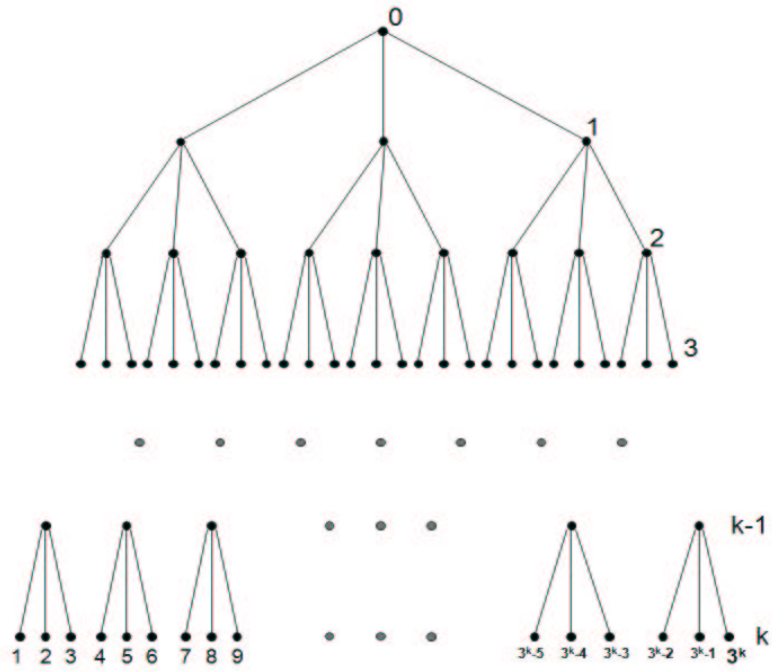

Figure 1: The computational tree of a non-deterministic Turing machine $\mathcal{M}_N$ having the non-deterministic degree $d = 3$

As for a deterministic Turing machine (see (14)), the behavior of $\mathcal{M}_N$ is specified by its transition function δ_N and consists of a sequence of computational steps. In each step, given the current state of the machine and the symbol it is reading on the tape, the transition function δ_N returns (if it is defined for these inputs) a set of triplets each of which specifies: (i) a symbol $\gamma \in \Gamma$ to write on the cell of the tape under the head; (ii) the move of the tape (L for one cell left, R for one cell right, N for no move); (iii) the next state $q \in Q$ of the Machine. Thus, in each computational step, the machine can *non-deterministically* execute different computations, one for each triple returned by the transition function.

An important characteristic of a non-deterministic Turing machine (see, e.g., [2]) is its non-deterministic degree

$$d = \nu(\mathcal{M}_N) = \max_{q \in Q-F, \gamma \in \Gamma} |\delta_N(q,\gamma)|$$

defined as the maximal number of different configurations reachable in a single computational step starting from a given configuration. The behavior of the machine can be then represented as a tree whose branches are the computations that the machine can execute starting from the initial configuration represented by the node 0 and nodes of the tree at the levels 1, 2, etc. represent subsequent configurations of the machine.

Let us consider an example shown in Fig. 1 where a non-deterministic machine $\mathcal{M}_N$ having the non-deterministic degree $d = 3$ is presented. The depth of the computational tree is equal to k. In this example, it is supposed that the computational tree of $\mathcal{M}_N$ is complete (i.e., each node has exactly d children). Then, obviously, the computational tree of $\mathcal{M}_N$ has $d^k = 3^k$ leaf nodes.

An important result for the classic theory on Turing machines (see e.g., [2]) is that for any non-deterministic Turing machine $\mathcal{M}_N$ there exists an equivalent deterministic Turing machine $\mathcal{M}_D$. Moreover, if the depth of the computational tree generated by $\mathcal{M}_N$ is equal to k, then for simulating $\mathcal{M}_N$, the deterministic machine $\mathcal{M}_D$ will execute at most

$$K_{\mathcal{M}_D} = \sum_{j=0}^{k} jd^j = O(kd^k)$$

computational steps.

Intuitively, for simulating $\mathcal{M}_N$, the deterministic Turing machine $\mathcal{M}_D$ executes a breadth-first visit of the computational tree of $\mathcal{M}_N$. If we consider the example from Fig. 1 with $k = 3$, then the computational tree of $\mathcal{M}_N$ has $d^k = 27$ leaf nodes and $d^k = 27$ computational paths consisting of $k = 3$ branches (i.e., computational steps). Then, the tree contains $d^{k-1} = 9$ computational paths consisting of $k-1 = 2$ branches and $d^{k-2} = 3$ computational paths consisting of $k-2 = 1$ branches. Thus, for simulating all the possible computations of $\mathcal{M}_N$, i.e., for complete visiting the computational tree of $\mathcal{M}_N$ and considering all the possible computational paths of j computational steps for each $0 \leqslant j \leqslant k$, the deterministic Turing machine $\mathcal{M}_D$ will execute $K_{\mathcal{M}_D}$ steps. In particular, if $\mathcal{M}_N$ reaches a final configuration (e.g., it accepts a string) in $k \geqslant 0$ steps and if $\mathcal{M}_D$ could consider only the d^k computational paths which consist of k computational steps, it will executes at most kd^k steps for reaching this configuration.

These results show an exponential growth of the time required for reaching a final configuration by the deterministic Turing machine $\mathcal{M}_D$ with respect to the time required by the non-deterministic Turing machine $\mathcal{M}_N$, assuming that the time required for both machines for a single step is the same. However, in the classic theory on Turing machines it is not known if there is a more efficient simulation of $\mathcal{M}_N$. In other words, it is an important and open problem of Computer Science theory to demonstrate that it is not possible to simulate a non-deterministic Turing machine by a deterministic Turing machine with a sub-exponential numbers of steps.

Let us now return to the new mathematical language. Since the main interest to machines (18) is related to their theoretical properties, hereinafter we start by a comparison of imaginary deterministic Turing machines, $\mathcal{T}^I$, with imaginary machines $\mathcal{M}_N$ from (18) denoted as $\mathcal{T}^{I\mathcal{N}}$. Physical machines $\mathcal{T}^{\mathcal{P}}$ and $\mathcal{T}^{\mathcal{PN}}$ are considered at the end of this section.

Due to the analysis made in Section 4.2, we should choose the accuracy (granularity) of processes of observation of both machines, $\mathcal{T}^I$ and $\mathcal{T}^{I\mathcal{N}}$. In order to

be close as much as possible to the traditional results, we consider an application of the transition function of the machine as our observation granularity. With respect to $\mathcal{T}^{I\mathcal{N}}$ this means that the nodes of the computational tree are observed. With respect to $\mathcal{T}^{I}$ we consider sequences of such nodes. For both cases the initial configuration is not observed, i.e., we start our observations from level 1 of the computational tree.

This choice of the observation granularity is particularly attractive due to its accordance with the traditional definitions of Turing machines (see definitions (14) and (18)). A more fine granularity of observations allowing us to follow internal operations of the machines can be also chosen but is not so convenient. In fact, such an accuracy would mix internal operations of the machines with operations of the algorithm that is executed. A coarser granularity could be considered, as well. For instance, we could define as a computational step two consecutive applications of the transition function of the machine. However, in this case we do not observe all the nodes of the computational tree. As a consequence, we could miss some results of the computation as the machine could reach a final configuration before completing an observed computational step and we are not able to observe when and on which configuration the machine stopped. Then, fixed the chosen level of granularity the following result holds immediately.

Theorem 5.1. *(i) With the chosen level of granularity no more than ① computational steps of the machine $\mathcal{T}^{I}$ can be observed in a sequence. (ii) In order to give possibility to observe at least one computational path of the computational tree of $\mathcal{T}^{I\mathcal{N}}$ from the level 1 to the level k, the depth, $k \geq 1$, of the computational tree cannot be larger than grossone, i.e., $k \leq ①$.*

Proof. Both results follow from the analysis made in Section 4.2 and Theorem 3.1. □

Corollary 5.1. *Suppose that d is the non-deterministic degree of $\mathcal{T}^{I\mathcal{N}}$ and S is the number of leaf nodes of the computational tree with a depth k representing the possible results of the computation of $\mathcal{T}^{I\mathcal{N}}$. Then it is not possible to observe all S possible results of the computation of $\mathcal{T}^{I\mathcal{N}}$ if the computational tree of $\mathcal{T}^{I\mathcal{N}}$ is complete and $d^k > ①$.*

Proof. For the number of leaf nodes of the tree, S, of a generic non-deterministic Turing machine $\mathcal{T}^{I\mathcal{N}}$ the estimate $S \leq d^k$ holds. In particular, $S = d^k$ if the computational tree is complete, that is our case. On the other hand, it follows from Theorem 3.1 that any sequence of observations cannot have more than grossone elements. As a consequence, the same limitation holds for the sequence of observations of the leaf nodes of the computational tree. This means that we are not able to observe all the possible results of the computation of our non-deterministic Turing machine $\mathcal{T}^{I\mathcal{N}}$ if $d^k > ①$. □

Corollary 5.2. *Any sequence of observations of the nodes of the computational tree of a non-deterministic Turing machine $\mathcal{T}^{I\mathcal{N}}$ cannot observe all the nodes of the tree if the number of nodes N is such that $N > ①$.*

Proof. The corollary follows from Theorems 3.1, 5.1, and Corollary 5.1. □

These results lead to the following theorem again under the same assumption about the chosen level of granularity of observations, i.e., the nodes of the computational tree of $\mathcal{T}^{I\mathcal{N}}$ representing configurations of the machine are observed.

Theorem 5.2. *Given a non-deterministic Turing machine $\mathcal{T}^{I\mathcal{N}}$ with a depth, k, of the computational tree and with a non-deterministic degree d such that*

$$\frac{d(kd^{k+1}-(k+1)d^k+1)}{(d-1)^2} \leqslant ①, \tag{19}$$

then there exists an equivalent deterministic Turing machine $\mathcal{T}^I$ which is able to simulate $\mathcal{T}^{I\mathcal{N}}$ and can be observed.

Proof. For simulating $\mathcal{T}^{I\mathcal{N}}$, the deterministic machine $\mathcal{T}^I$ executes a breadth-first visit of the computational tree of $\mathcal{T}^{I\mathcal{N}}$. In this computational tree, whose depth is $1 \leqslant k \leqslant ①$, each node has, by definition, a number of children c where $0 \leqslant c \leqslant d$. Let us suppose that the tree is complete, i.e., each node has $c = d$ children. In this case the tree has d^k leaf nodes and d^j computational paths of length j for each $1 \leqslant j \leqslant k$. Thus, for simulating all the possible computations of $\mathcal{T}^{I\mathcal{N}}$, i.e., for a complete visiting the computational tree of $\mathcal{T}^{I\mathcal{N}}$ and considering all the possible computational paths consisting of j computational steps for each $1 \leqslant j \leqslant k$, the deterministic machine $\mathcal{T}^I$ will execute

$$K_{\mathcal{T}^I} = \sum_{j=1}^{k} jd^j \tag{20}$$

steps (note that if the computational tree of $\mathcal{T}^{I\mathcal{N}}$ is not complete, $\mathcal{T}^I$ will execute less than $K_{\mathcal{T}^I}$). Due to Theorems 3.1 and 5.1, and Corollary 5.2, it follows that in order to prove the theorem it is sufficient to show that under conditions of the theorem it follows that

$$K_{\mathcal{T}^I} \leqslant ①. \tag{21}$$

To do this let us use the well known formula

$$\sum_{j=0}^{k} d^j = \frac{d^{k+1}-1}{d-1}, \tag{22}$$

and derive both parts of (22) with respect to d. As the result we obtain

$$\sum_{j=1}^{k} jd^{j-1} = \frac{kd^{k+1}-(k+1)d^k+1}{(d-1)^2}. \tag{23}$$

Notice now that by using (20) it becomes possible to represent the number $K_{\mathcal{T}^I}$ as

$$K_{\mathcal{T}^I} = \sum_{j=1}^{k} jd^j = d\sum_{j=1}^{k} jd^{j-1}.$$

This representation together with (23) allow us to write

$$K_{\mathcal{T}^I} = \frac{d(kd^{k+1} - (k+1)d^k + 1)}{(d-1)^2} \tag{24}$$

Due to assumption (19), it follows that (21) holds. This fact concludes the proof of the theorem. □

Corollary 5.3. *Suppose that the length of the input sequence of symbols of a non-deterministic Turing machine $\mathcal{T}^{I\mathcal{N}}$ is equal to a number n and $\mathcal{T}^{I\mathcal{N}}$ has a complete computational tree with the depth k such that $k = n^l$, i.e., polynomially depends on the length n. Then, if the values d, n, and l satisfy the following condition*

$$\frac{d(n^l d^{n^l+1} - (n^l+1)d^{n^l} + 1)}{(d-1)^2} \leqslant ①, \tag{25}$$

then: (i) there exists a deterministic Turing machine $\mathcal{T}^I$ that can be observed and able to simulate $\mathcal{T}^{I\mathcal{N}}$; (ii) the number, $K_{\mathcal{T}^I}$, of computational steps required to a deterministic Turing machine $\mathcal{T}^I$ to simulate $\mathcal{T}^{I\mathcal{N}}$ for reaching a final configuration exponentially depends on n.

Proof. The first assertion follows immediately from theorem 5.2. Let us prove the second assertion. Since the computational tree of $\mathcal{T}^{I\mathcal{N}}$ is complete and has the depth k, the corresponding deterministic Turing machine $\mathcal{T}^I$ for simulating $\mathcal{T}^{I\mathcal{N}}$ will execute $K_{\mathcal{T}^I}$ steps where $K_{\mathcal{T}^I}$ is from (21). Since condition (25) is satisfied for $\mathcal{T}^{I\mathcal{N}}$, we can substitute $k = n^l$ in (24). As the result of this substitution and (25) we obtain that

$$K_{\mathcal{T}^I} = \frac{d(n^l d^{n^l+1} - (n^l+1)d^{n^l} + 1)}{(d-1)^2} \leqslant ①, \tag{26}$$

i.e., the number of computational steps required to the deterministic Turing machine $\mathcal{T}^I$ to simulate the non-deterministic Turing machine $\mathcal{T}^{I\mathcal{N}}$ for reaching a final configuration is $K_{\mathcal{T}^I} \leqslant ①$ and this number exponentially depends on the length of the sequence of symbols provided as input to $\mathcal{T}^{I\mathcal{N}}$. □

Results described in this section show that the introduction of the new mathematical language including grossone allows us to perform a more subtle analysis with respect to traditional languages and to introduce in the process of this analysis the figure of the researcher using this language (more precisely, to emphasize the presence of the researcher in the process of the description of automatic computations). These results show that there exist limitations for simulating non-deterministic Turing machines by deterministic ones. These limitations can be viewed now thanks to the possibility (given because of the introduction of the new numeral ①) to observe final points of sequential processes for both cases of finite and infinite processes.

Theorems 5.1, 5.2, and their corollaries show that the discovered limitations and relations between deterministic and non-deterministic Turing machines have

strong links with our mathematical abilities to describe automatic computations and to construct models for such descriptions. Again, as it was in the previous cases studied in this paper, there is no contradiction with the traditional results because both approaches give results that are correct with respect to the languages used for the respective descriptions of automatic computations.

We conclude this section by the note that analogous results can be obtained for physical machines $\mathcal{T}^{\mathcal{P}}$ and $\mathcal{T}^{\mathcal{PN}}$, as well. In the case of imaginary machines, the possibility of observations was limited by the mathematical languages. In the case of physical machines they are limited also by technical factors (we remind again the analogy: the possibilities of observations of physicists are limited by their instruments). In any given moment of time the maximal number of iterations, K_{max}, that can be executed by physical Turing machines can be determined. It depends on the speed of the fastest machine $\mathcal{T}^{\mathcal{P}}$ available at the current level of development of the humanity, on the capacity of its memory, on the time available for simulating a non-deterministic machine, on the numeral systems known to human beings, etc. Together with the development of technology this number will increase but it will remain finite and fixed in any given moment of time. As a result, theorems presented in this section can be re-written for $\mathcal{T}^{\mathcal{P}}$ and $\mathcal{T}^{\mathcal{PN}}$ by substituting grossone with K_{max} in them.

6 Conclusion

The problem of mathematical descriptions of automatic computations (the concept of the Turing machine has been used as a model of a device executing such computations) has been considered in this paper from several points of view. First, the problem has been studied using a new methodology emphasizing in a strong form the presence in the process of the description of automatic computations of the researcher who describes a computational device and its properties. The role of the philosophical triad – the researcher, the object of investigation, and tools used to observe the object – has been emphasized in the study. A deep investigation has been performed on the interrelations that arise between mechanical computations themselves and their mathematical descriptions when a human (the researcher) starts to describe a Turing machine (the object of the study) by different mathematical languages (the instruments of investigation).

Along with traditional mathematical languages using such concepts as 'enumerable sets' and 'continuum' to describe the potential of automatic computations, a language introduced recently and the corresponding computational methodology allowing one to measure the number of elements of different infinite sets have been used in this paper. It has been emphasized that mathematical descriptions obtained by using different languages depict the object of the study – the Turing machine – in different ways. It has been established that the obtained descriptions, even though in certain cases they give different answers to the same questions, do not contradict each other. All of them are correct with respect to the language used for the

observation and the description of the machines.

It has been established that there exists the relativity of mathematical descriptions of the object and there cannot be ever any assurance that a language chosen for the current description expresses the object in an absolutely correct and complete way. A richer language allows the researcher to reflect better the properties of the studied object and a weaker language does this worse (however, this fact can be noticed only if a richer language is already known to the researcher). This situation is similar to the work with a microscope where, when we need a more precise or a more rough answer, we change the lens of our microscope. For instance, suppose that by using a weak lens A we see the object of observation as one black dot while by using a stronger lens B we see that the object of observation consists of two (smaller) black dots. Thus, we have two different answers: (i) the object consists of one dot; (ii) the object consists of two dots. Both answers are correct with respect to the lens used for the observation.

The new mathematical language applied in this study has allowed the authors to establish a number of results regarding sequential computations executed by the Turing machine and results regarding computable sequences produced by the machine. Deterministic and non-deterministic machines have been studied using both the traditional and the new languages. The obtained results have been compared and discussed.

It has been emphasized that all mathematical (and not only mathematical) languages (including the new one used in this study) have limited expressibilities. This fact leads to several important reflections. First, for any fixed language there always exist problems that cannot be formulated using it (these problems often can be seen when a new, sufficiently powerful for this purpose language is invented). Second, there always exist problems such that questions regarding these problems can be formulated in a language but this language is too weak to express the desired answer or the accuracy of the obtained answer is insufficient for the practical needs. Finally, in any given moment of time for each concrete problem there exists a finite number of languages that can be used to attack the problem. Then, the most powerful language among them defines computational bounds for the problem that exist both for physical and imaginary Turing machines (i.e., in both cases when a maximal finite or a maximal infinite number of iterations is considered).

References

[1] A. Adamatzky, B. De Lacy Costello, and T. Asai. *Reaction-diffusion computers*. Elsevier, Amsterdam, 2005.

[2] G. Ausiello, F. D'Amore, and G. Gambosi. *Linguaggi, modelli, complessità*. Franco Angeli Editore, Milan, 2 edition, 2006.

[3] G. Cantor. *Contributions to the founding of the theory of transfinite numbers*. Dover Publications, New York, 1955.

[4] J.B. Carroll, editor. *Language, Thought, and Reality: Selected Writings of Benjamin Lee Whorf*. MIT Press, 1956.

[5] A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.

[6] S. Barry Cooper. *Computability Theory*. Chapman Hall/CRC, 2003.

[7] M. Davis. *Computability & Unsolvability*. Dover Publications, New York, 1985.

[8] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.

[9] P. Gordon. Numerical cognition without words: Evidence from Amazonia. *Science*, 306(15 October):496–499, 2004.

[10] J.J. Gumperz and S.C. Levinson, editors. *Rethinking Linguistic Relativity*. Cambridge University Press, Cambridge, 1996.

[11] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading Mass., 1st edition, 1979.

[12] K.E. Iverson. Notation as a tool of thought. *Communications of the ACM*, 23:444–465, August 1980.

[13] S.C. Kleene. *Introduction to metamathematics*. D. Van Nostrand, New York, 1952.

[14] A.N. Kolmogorov. On the concept of algorithm. *Uspekhi Mat. Nauk*, 8(4):175–176, 1953.

[15] A.N. Kolmogorov and V.A. Uspensky. On the definition of algorithm. *Uspekhi Mat. Nauk*, 13(4):3–28, 1958.

[16] J.A. Lucy. *Grammatical Categories and Cognition: A Case Study of the Linguistic Relativity Hypothesis*. Cambridge University Press, Cambridge, 1992.

[17] A.A. Markov Jr. and N.M. Nagorny. *Theory of Algorithms*. FAZIS, Moscow, second edition, 1996.

[18] J.P. Mayberry. *The Foundations of Mathematics in the Theory of Sets*. Cambridge University Press, Cambridge, 2001.

[19] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2000.

[20] P. Pica, C. Lemer, V. Izard, and S. Dehaene. Exact and approximate arithmetic in an amazonian indigene group. *Science*, 306(15 October):499–503, 2004.

[21] E. Post. Finite combinatory processes – formulation 1. *Journal of Symbolic Logic*, 1:103–105, 1936.

[22] A. Robinson. *Non-standard Analysis*. Princeton Univ. Press, Princeton, 1996.

[23] E. Sapir. *Selected Writings of Edward Sapir in Language, Culture and Personality*. University of California Press, Princeton, 1958.

[24] Ya.D. Sergeyev. *Arithmetic of Infinity*. Edizioni Orizzonti Meridionali, CS, 2003.

[25] Ya.D. Sergeyev. *http://www.theinfinitycomputer.com*. 2004.

[26] Ya.D. Sergeyev. A new applied approach for executing computations with infinite and infinitesimal quantities. *Informatica*, 19(4):567–596, 2008.

[27] Ya.D. Sergeyev. *Computer system for storing infinite, infinitesimal, and finite quantities and executing arithmetical operations with them*. EU patent 1728149, 2009.

[28] Ya.D. Sergeyev. Evaluating the exact infinitesimal values of area of Sierpinski's carpet and volume of Menger's sponge. *Chaos, Solitons & Fractals*, 42(5):3042–3046, 2009.

[29] Ya.D. Sergeyev. Numerical computations and mathematical modelling with infinite and infinitesimal numbers. *Journal of Applied Mathematics and Computing*, 29:177–195, 2009.

[30] Ya.D. Sergeyev. Numerical point of view on Calculus for functions assuming finite, infinite, and infinitesimal values over finite, infinite, and infinitesimal domains. *Nonlinear Analysis Series A: Theory, Methods & Applications*, 71(12):e1688–e1707, 2009.

[31] Ya.D. Sergeyev. Counting systems and the First Hilbert problem. *Nonlinear Analysis Series A: Theory, Methods & Applications*, 72(3-4):1701–1708, 2010.

[32] A.M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of London Mathematical Society, series 2*, 42:230–265, 1936-1937.

[33] Žilinskas A. and Žilinskas J. Interval arithmetic based optimization in nonlinear regression. *Informatica*, 21(1):149–158, 2010.

[34] G.W. Walster. *Compiler Support of Interval Arithmetic With Inline Code Generation and Nonstop Exception Handling*. Tech. Report, Sun Microsystems, 2000.