Noname manuscript No.
(will be inserted by the editor)

Single-tape and Multi-tape Turing Machines through the lens of the Grossone methodology*

Yaroslav D. Sergeyev · **Alfredo Garro**

Received: date / Accepted: date

Abstract The paper investigates how the mathematical languages used to describe and to observe automatic computations influence the accuracy of the obtained results. In particular, we focus our attention on Single and Multi-tape Turing machines which are described and observed through the lens of a new mathematical language which is strongly based on three methodological ideas borrowed from Physics and applied to Mathematics, namely: the distinction between the object (we speak here about a mathematical object) of an observation and the instrument used for this observation; interrelations holding between the object and the tool used for the observation; the accuracy of the observation determined by the tool. Results of the observation executed by the traditional and new languages are compared and discussed.

Keywords Theory of automatic computations · Observability of Turing machines · Relativity of mathematical languages · Infinite sequences · Infinite sets

* This research was supported by the project "High accuracy supercomputations and solving global optimization problems using the information approach" of the Russian Federal Program "Scientists and Educators in Russia of Innovations", grant number 14.B37.21.0878.

Yaroslav D. Sergeyev
Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, Rende (CS), Italy
N.I. Lobatchevsky State University, Nizhni Novgorod, Russia
Istituto di Calcolo e Reti ad Alte Prestazioni, C.N.R., Rende (CS), Italy
Via P. Bucci 42C, 87036 Rende (CS), Italy
Tel.: +39-0984-494855
Fax: +39-0984-494713
E-mail: yaro@si.deis.unical.it

Alfredo Garro
Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria
Via P. Bucci 41C, 87036 Rende (CS), Italy
Tel.: +39-0984-494795
Fax: +39-0984-494713
E-mail: alfredo.garro@unical.it

1 Introduction

Since the beginning of the last century, the fundamental nature of the concept of *automatic computations* attracted a great attention of mathematicians and computer scientists (see [5,15–17,23,24,28,43]). The first studies had as their reference context the David Hilbert programme, and as their reference language that introduced by Georg Cantor [4]. These approaches lead to different mathematical models of computing machines (see [2,7,10]) that, surprisingly, were discovered to be equivalent (e.g., anything computable in the λ-calculus is computable by a Turing machine). Moreover, these results, and expecially those obtained by Alonzo Church, Alan Turing [5,11,43] and Kurt Gödel, gave fundamental contributions to demonstrate that David Hilbert programme, which was based on the idea that all of the Mathematics could be precisely axiomatized, cannot be realized.

In spite of this fact, 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. Usually, when it is necessary to define a concept or an object, logicians try to introduce a number of axioms describing the object in the absolutely best way. However, it is not clear how to reach this absoluteness; indeed, when we describe a mathematical object or a 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, the continuous 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 [11]).

Starting from these considerations, 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 [43]) whereas emerging computational paradigms (see, e.g., [1,26,45,47]) are not considered here. In particular, we focus our attention on different kinds of Turing machines by enriching and extending the results presented in [42].

The point of view presented in this paper uses strongly three methodological ideas borrowed from Physics and applied to Mathematics, namely: the distinction between the object (we speak here about a mathematical object) of an observation and the instrument used for this observation; interrelations holding between the object and the tool used for this observation; the accuracy of the observation determined by the tool.

The main attention is dedicated to numeral systems[1] that we use to write down numbers, functions, models, etc. and that are among our tools of investigation of mathematical and physical objects. It is shown that numeral systems strongly influence our capabilities to describe both the mathematical and physical worlds. A new numeral system introduced in [31,33,38]) for performing computations with infinite and infinitesimal quantities is used for the observation of mathematical objects and studying Turing machines. The new methodology is based on the principle 'The part is less than the whole' introduced by Ancient Greeks and observed in practice. It is applied to all sets and processes (finite and infinite) and all numbers (finite, infinite, and infinitesimal).

In order to see the place of the new approach in the historical panorama of ideas dealing with infinite and infinitesimal, see [20,21,36,37,42]. The new methodology has been successfully applied for studying a number of applications: percolation (see [14,44]), Euclidean and hyperbolic geometry (see [22,30]), fractals (see [32,34,41, 44]), numerical differentiation and optimization (see [8,35,39,49]), infinite series (see [36,40,48]), the first Hilbert problem (see [37]), and cellular automata (see [9]).

The rest of the paper is structured as follows. In Section 2, Single and Multi-tape Turing machines are introduced along with "classical" results concerning their computational power and related equivalences; in Section 3 a brief introduction to the new language and methodology is given whereas their exploitation for analyzing and observing the different types of Turing machines is discussed in Section 4. It shows that the new approach allows us to observe Turing machines with a higher accuracy giving so the possibility to better characterize and distinguish machines which are equivalent when observed within the classical framework. Finally, Section 5 concludes the paper.

2 Single and Multi-tape Turing Machines

The Turing machine is one of the simple abstract computational devices that can be used to model computational processes and investigate the limits of computability. In the following Subsections 2.1 and 2.2, Single and Multi-tape Turing machines will be described along with important classical results concerning their computational power and related equivalences.

2.1 Single Tape Turing Machines

A Turing Machine (see, e.g., [13,43]) can be defined as a 7-tuple

$$\mathcal{M} = \left\langle Q, \Gamma, \overline{b}, \Sigma, q_0, F, \delta \right\rangle, \tag{1}$$

[1] We are reminded that a *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 '7', 'seven', and 'VII' are different numerals, but they all represent the same number.

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; $\delta : \{Q - F\} \times \Gamma \mapsto Q \times \Gamma \times \{R, L, N\}$ 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 $\bar{b} \in \Gamma$ is the only symbol allowed to occur on the tape infinitely often; (ii) a *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.

Starting from the definition of Turing Machine introduced above, classical results (see, e.g., [2]) aim at showing that different machines in terms of provided tape and alphabet have the same computational power, i.e., they are able to execute the same computations. In particular, two main results are reported below in an informal way.

Given a Turing Machine $\mathcal{M} = \{Q, \Gamma, \bar{b}, \Sigma, q_0, F, \delta\}$, which is supplied with an infinite tape, it is always possible to define a Turing Machine $\mathcal{M}' = \{Q', \Gamma', \bar{b}, \Sigma', q_0', F', \delta'\}$ which is supplied with a semi-infinite tape (e.g., a tape with a left boundary) and is equivalent to $\mathcal{M}$, i.e., is able to execute all the computations of $\mathcal{M}$.

Given a Turing Machine $\mathcal{M} = \{Q, \Gamma, \bar{b}, \Sigma, q_0, F, \delta\}$, it is always possible to define a Turing Machine $\mathcal{M}' = \{Q', \Gamma', \bar{b}, \Sigma', q_0', F', \delta'\}$ with $|\Sigma'| = 1$ and $\Gamma' = \Sigma' \cup \{\bar{b}\}$, which is equivalent to $\mathcal{M}$, i.e., is able to execute all the computations of $\mathcal{M}$.

It should be mentioned that these results, together with the usual conclusion regarding the equivalences of Turing machines, can be interpreted in the following, less obvious, way: they show that when we observe Turing machines by exploiting the classical framework we are not able to distinguish, from the computational point of view, Turing machines which are provided with alphabets having different number of symbols and/or different kind of tapes (infinite or semi-infinite) (see [42] for a detailed discussion).

2.2 Multi-tape Turing Machines

Let us consider a variant of the Turing Machine defined in (1) where a machine is equipped with multiple tapes that can be simultaneously accessed and updated through multiple heads (one per tape). These machines can be used for a more direct and intuitive resolution of different kind of computational problems. As an example, in checking if a string is palindrome it can be useful to have two tapes on which represent the input string so that the verification can be efficiently performed by reading a tape from left to right and the other one from right to left.

Moving towards a more formal definition, a k-tapes, $k \geq 2$, Turing machine (see [13]) can be defined (cf. (1)) as a 7-tuple

$$\mathcal{M}_K = \left\langle Q, \Gamma, \bar{b}, \Sigma, q_0, F, \delta^{(k)} \right\rangle, \tag{2}$$

where $\Sigma = \bigcup_{i=1}^{k} \Sigma_i$ is given by the union of the symbols in the k input/output alphabets $\Sigma_1, \ldots, \Sigma_k$; $\Gamma = \Sigma \cup \{\bar{b}\}$ where $\bar{b}$ is a symbol called blank; Q is a finite and not empty set of states; $q_0 \in Q$ is the initial state; $F \subseteq Q$ is the set of final states; $\delta^{(k)} : \{Q - F\} \times \Gamma_1 \times \cdots \times \Gamma_k \mapsto Q \times \Gamma_1 \times \cdots \times \Gamma_k \times \{R, L, N\}^k$ is a partial function called the transition function, where $\Gamma_i = \Sigma_i \cup \{\bar{b}\}, i = 1, \ldots, k$, L means left, R means right, and N means no move.

This definition of $\delta^{(k)}$ means that the machine executes a transition starting from an internal state q_i and with the k heads (one for each tape) above the characters $a_{i1}, \ldots, a_{ik}$, i.e., if $\delta^{(k)}(q_1, a_{i1}, \ldots, a_{ik}) = (q_j, a_{j_1}, \ldots, a_{j_k}, z_{j_1}, \ldots, z_{j_k})$ the machine goes in the new state q_j, write on the k tapes the characters $a_{j_1}, \ldots, a_{j_k}$ respectively, and moves each of its k heads left, right or no move, as specified by the $z_{j_l} \in \{R, L, N\}, l = 1, \ldots, k$.

A machine can adopt for each tape a different alphabet, in any case, for each tape, as for the Single-tape Turing machines, the minimum portion containing characters distinct from $\bar{b}$ is usually represented. In general, a typical configuration of a Multi-tape machine consists of a read-only input tape, several read and write work tapes, and a write-only output tape, with the input and output tapes accessible only in one direction. In the case of a k-tapes machine, the instant configuration of the machine, as for the Single-tape case, must describe the internal state, the contents of the tapes and the positions of the heads of the machine.

More formally, for a k-tapes Turing machine $\mathcal{M}_K = \left\langle Q, \Gamma, \bar{b}, \Sigma, q_0, F, \delta^{(k)} \right\rangle$ with $\Sigma = \bigcup_{i=1}^{k} \Sigma_i$ (see 2) a configuration of the machine is given by:

$$q\#\alpha_1 \uparrow \beta_1 \# \alpha_2 \uparrow \beta_2 \# \ldots \# \alpha_k \uparrow \beta_k, \tag{3}$$

where $q \in Q$; $\alpha_i \in \Sigma_i \Gamma_i^* \cup \{\varepsilon\}$ and $\beta_i \in \Gamma_i^* \Sigma_i \cup \{\bar{b}\}$. A configuration is *final* if $q \in F$.

The *starting* configuration usually requires the input string x on a tape, e.g., the first tape so that $x \in \Sigma_1^*$, and only $\bar{b}$ symbols on all the other tapes. However, it can be useful to assume that, at the beginning of a computation, these tapes have a starting symbol $Z_0 \notin \Gamma = \bigcup_{i=1}^{k} \Gamma_i$. Therefore, in the initial configuration the head on the first tape will be on the first character of the input string x, whereas the heads on the other tapes will observe the symbol Z_0, more formally, by re-placing $\Gamma_i = \Sigma_i \cup \{\bar{b}, Z_0\}$ in all the previous definition, a configuration $q\#\alpha_1 \uparrow \beta_1 \# \alpha_2 \uparrow \beta_2 \# \ldots \# \alpha_k \uparrow \beta_k$ is an *initial configuration* if $\alpha_i = \varepsilon, i = 1, \ldots, k, \beta_1 \in \Sigma_1^*, \beta_i = Z_0, i = 2, \ldots, k$ and $q = q_0$.

The application of the transition function $\delta^{(k)}$ at a machine configuration (c.f. (3)) defines a *computational step* of a Multi-tape Turing Machine. The set of computational steps which bring the machine from the initial configuration into a final configuration defines the *computation* executed by the machine. As an example, the computation of a Multi-tape Turing machine $\mathcal{M}_K$ which computes the function $f_{\mathcal{M}_K}(x)$ can be represented as follows:

$$q_0\# \uparrow x\# \uparrow Z_0\# \ldots \# \uparrow Z_0 \overset{\rightarrow}{\mathcal{M}_K} q\# \uparrow x\# \uparrow f_{\mathcal{M}_K}(x)\# \uparrow \bar{b}\# \ldots \# \uparrow \bar{b} \tag{4}$$

where $q \in F$ and $\vec{\mathcal{M}}_K$ indicates the transition among machine configurations.

It is worth noting that, although the k-tapes Turing Machine can be used for a more direct resolution of different kind of computational problems, in the classical framework it has the same computational power of the Single-tape Turing machine. More formally, given a Multi-tape Turing Machine it is always possible to define a Single-tape Turing Machine which is able to fully simulate its behavior and therefore to completely execute its computations. In particular, the Single-tape Turing Machines adopted for the simulation use a particular kind of the tape which is divided into tracks (multi-track tape). In this way, if the tape has m tracks, the head is able to access (for reading and/or writing) all the m characters on the tracks during a single operation. If for the m tracks the alphabets $\Gamma_1, \dots \Gamma_m$ are adopted respectively, the machine alphabet Γ is such that $|\Gamma| = |\Gamma_1 \times \cdots \times \Gamma_m|$ and can be defined by an injective function from the set $\Gamma_1 \times \cdots \times \Gamma_m$ to the set Γ; this function will associate the symbol $\bar{b}$ in Γ to the tuple $(\bar{b}, \bar{b}, \dots, \bar{b})$ in $\Gamma_1 \times \cdots \times \Gamma_m$. In general, the elements of Γ which correspond to the elements in $\Gamma_1 \times \cdots \times \Gamma_m$ can be indicated by $[a_{i1}, a_{i2}, \dots, a_{im}]$ where $a_{ij} \in \Gamma_j$.

By adopting this notation it is possible to demonstrate that given a k-tapes Turing Machine $\mathcal{M}_K = \{Q, \Gamma, \bar{b}, \Sigma, q_0, F, \delta^{(k)}\}$ it is always possible to define a Single-tape Turing Machine which is able to simulate t computational steps of $\mathcal{M}_K =$ in $O(t^2)$ transitions by using an alphabet with $O((2|\Gamma|)^k)$ symbols (see [2]).

The proof is based on the definition of a machine $\mathcal{M}' = \{Q', \Gamma', \bar{b}, \Sigma', q_0', F', \delta'\}$ with a Single-tape divided into $2k$ tracks (see [2]); k tracks for storing the characters in the k tapes of $\mathcal{M}_K$ and k tracks for signing through the marker $\downarrow$ the positions of the k heads on the k tapes of $\mathcal{M}_k$. As an example, this kind of tape can represent the content of each tapes of $\mathcal{M}_k$ and the position of each machine heads in its even and odd tracks respectively. As discussed above, for obtaining a Single-tape machine able to represent these $2k$ tracks, it is sufficient to adopt an alphabet with the required cardinality and define an injective function which associates a 2k-ple characters of a cell of the multi-track tape to a symbols in this alphabet.

The transition function $\delta^{(k)}$ of the k-tapes machine is given by $\delta^{(k)}(q_1, a_{i1}, \dots, a_{ik}) = (q_j, a_{j_1}, \dots, a_{j_k}, z_{j_1}, \dots, z_{j_k})$, with $z_{j_1}, \dots, z_{j_k} \in \{R, L, N\}$; as a consequence the corresponding transition function δ' of the Single-tape machine, for each transition specified by $\delta^{(k)}$ must individuate the current state and the position of the marker for each track and then write on the tracks the required symbols, move the markers and go in another internal state. For each computational step of $\mathcal{M}_K$, the machine $\mathcal{M}'$ must execute a sequence of steps for covering the portion of tapes between the two most distant markers. As in each computational step a marker can move at most of one cell and then two markers can move away each other at most of two cells, after t steps of $\mathcal{M}_K$ the markers can be at most $2t$ cells distant, thus if $\mathcal{M}_K$ executes t steps, $\mathcal{M}'$ executes at most: $2\sum_{i=1}^{t} i = t^2 + t = O(t^2)$ steps.

Moving to the cost of the simulation in terms of the number of required characters for the alphabet of the Single-tape machine, we recall that $|\Gamma_1| = |\Sigma_1| + 1$ and that $|\Gamma_i| = |\Sigma_i| + 2$ for $2 \leq i \leq k$. So by multiplying the cardinalities of these alphabets we obtain that: $|\Gamma'| = 2^k(|\Sigma_1| + 1)\prod_{i=2}^{k}(|\Sigma_i| + 2) = O((2max_{1 \leq i \leq k}|\Gamma_i|)^k)$.

3 The Grossone Methodology

In this section, we give just a brief introduction to the methodology of the new approach [31,33] dwelling only on the issues directly related to the subject of the paper. This methodology will be used in Section 4 to study Turing machines and to obtain some more accurate results with respect to those obtainable by using the traditional framework [5,43].

In order to start, let us remind that numerous trials have been done during the centuries to evolve existing numeral systems in such a way that numerals representing infinite and infinitesimal numbers could be included in them (see [3,4,6,18,19,25, 29,46]). Since new numeral systems appear very rarely, in each concrete historical period their significance for Mathematics is very often underestimated (especially by pure mathematicians). In order to illustrate their importance, let us remind the Roman numeral system that does not allow one to express zero and negative numbers. In this system, the expression III-X is an indeterminate form. As a result, before appearing the positional numeral system and inventing zero mathematicians were not able to create theorems involving zero and negative numbers and to execute computations with them.

There exist numeral systems that are even weaker than the Roman one. They seriously limit their users in executing computations. Let us recall a study published recently in *Science* (see [12]). It describes a primitive tribe living in Amazonia (Pirahã). These people use a very simple numeral system 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'. Using their weak numeral system Pirahã are not able to see, for instance, numbers 3, 4, 5, and 6, 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.

In the context of the present paper, it is very important that the weakness of Pirahã's numeral system leads them to such results as

$$\text{'many'} + 1 = \text{'many'}, \qquad \text{'many'} + 2 = \text{'many'}, \tag{5}$$

which are very familiar to us in the context of views on infinity used in the traditional calculus

$$\infty + 1 = \infty, \qquad \infty + 2 = \infty. \tag{6}$$

The arithmetic of Pirahã involving the numeral 'many' has also a clear similarity with the arithmetic proposed by Cantor for his Alephs[2]:

$$\aleph_0 + 1 = \aleph_0, \qquad \aleph_0 + 2 = \aleph_0, \qquad \aleph_1 + 1 = \aleph_1, \qquad \aleph_1 + 2 = \aleph_1. \tag{7}$$

Thus, the modern mathematical numeral systems allow us to distinguish a larger quantity of finite numbers with respect to Pirahã but give results that are similar to

[2] This similarity becomes even more pronounced if one considers another Amazonian tribe – Mundurukú (see [27]) – who 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 using the rules that are very similar to ones used by Cantor to operate with $\aleph_0$ and $\aleph_1$, respectively.

those of Pirahã when we speak about infinite quantities. This observation leads us to the following idea: *Probably our difficulties in working with infinity is not connected to the nature of infinity itself but is a result of inadequate numeral systems that we use to work with infinity, more precisely, to express infinite numbers.*

The approach developed in [31,33,38] proposes a numeral system that uses the same numerals for several different purposes for dealing with infinities and infinitesimals: in Analysis for working with functions that can assume different infinite, finite, and infinitesimal values (functions can also have derivatives assuming different infinite or infinitesimal values); for measuring infinite sets; for indicating positions of elements in ordered infinite sequences; in probability theory, etc. (see [8,9,14,22, 30,32,34–37,39–41,44,48,49]). It is important to emphasize that the new numeral system avoids situations of the type (5)–(7) providing results ensuring that if a is a numeral written in this system then for any a (i.e., a can be finite, infinite, or infinitesimal) it follows $a+1>a$.

The new numeral system works as follows. A new infinite unit of measure expressed by the numeral ① called *grossone* is introduced as the number of elements of the set, $\mathbb{N}$, of natural numbers. Concurrently with the introduction of grossone in the mathematical language all other symbols (like ∞, Cantor's ω, $\aleph_0, \aleph_1, ...$, etc.) traditionally used to deal with infinities and infinitesimals are excluded from the language because grossone and other numbers constructed with its help not only can be used instead of all of them but can be used with a higher accuracy[3]. Grossone is introduced by describing its properties postulated by the Infinite Unit Axiom (see [33, 38]) added to axioms for real numbers (similarly, in order to pass from the set, $\mathbb{N}$, of natural numbers to the set, $\mathbb{Z}$, of integers a new element – zero expressed by the numeral 0 – is introduced by describing its properties).

The new numeral ① allows us to construct different numerals expressing different infinite and infinitesimal numbers and to execute computations with them. Let us give some examples. For instance, in Analysis, indeterminate forms are not present and, for example, the following relations hold for ① and $①^{-1}$ (that is infinitesimal), as for any other (finite, infinite, or infinitesimal) number expressible in the new numeral system

$$0\cdot①=①\cdot 0=0,\quad ①-①=0,\quad \frac{①}{①}=1,\quad ①^0=1,\quad 1^{①}=1,\quad 0^{①}=0, \tag{8}$$

$$0\cdot①^{-1}=①^{-1}\cdot 0=0,\quad ①^{-1}>0,\quad ①^{-2}>0,\quad ①^{-1}-①^{-1}=0, \tag{9}$$

$$\frac{①^{-1}}{①^{-1}}=1,\quad \frac{①^{-2}}{①^{-2}}=1,\quad (①^{-1})^0=1,\quad ①\cdot①^{-1}=1,\quad ①\cdot①^{-2}=①^{-1}. \tag{10}$$

The new approach gives the possibility to develop a new Analysis (see [36]) where functions assuming not only finite values but also infinite and infinitesimal ones can be studied. For all of them it becomes possible to introduce a new notion of continuity that is closer to our modern physical knowledge. Functions assuming finite and infinite values can be differentiated and integrated.

[3] Analogously, when the switch from Roman numerals to the Arabic ones has been done, numerals X, V, I, etc. have been excluded from records using Arabic numerals.

By using the new numeral system it becomes possible to measure certain infinite sets and to see, e.g., that the sets of even and odd numbers have $①/2$ elements each. The set, $\mathbb{Z}$, of integers has $2①+1$ elements ($①$ positive elements, $①$ negative elements, and zero). Within the countable sets and sets having cardinality of the continuum (see [20,37,38]) it becomes possible to distinguish infinite sets having different number of elements expressible in the numeral system using grossone and to see that, for instance,

$$\frac{①}{2} < ①-1 < ① < ①+1 < 2①+1 < 2①^2-1 < 2①^2 < 2①^2+1 <$$

$$2①^2+2 < 2^{①}-1 < 2^{①} < 2^{①}+1 < 10^{①} < ①^{①}-1 < ①^{①} < ①^{①}+1. \qquad (11)$$

Another key notion for our study of Turing machines is that of infinite sequence. 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 that the removal of the elements from $\{a_n\}$ can be directly observed, 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 $\{a_n\}$.

Let us study what happens when the new approach is used. 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. 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. In the finite case, a sequence $a_1, a_2, \ldots, a_n$ has n elements and we extend this definition directly to the infinite case saying that an infinite sequence $a_1, a_2, \ldots, a_n$ has n elements where n is expressed by an infinite numeral such that the operations with it satisfy the methodological Postulate 3. Then the following result (see [31,33]) holds. We reproduce here its proof for the sake of completeness.

Theorem 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. 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 [31,33]). 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; the following example clarifies these concepts.

Example 1 Let us consider the following three sequences:

$$\{a_n : ①\} = \{4, \quad 8, \quad \ldots \quad 4(① - 1), \quad 4①\}; \tag{12}$$

$$\{b_n : \frac{①}{2} - 1\} = \{4, \quad 8, \quad \ldots \quad 4(\frac{①}{2} - 2), \quad 4(\frac{①}{2} - 1)\}; \tag{13}$$

$$\{c_n : \frac{2①}{3}\} = \{4, \quad 8, \quad \ldots \quad 4(\frac{2①}{3} - 1), \quad 4\frac{2①}{3}\}. \tag{14}$$

The three sequences have $a_n = b_n = c_n = 4n$ but they are different because they have different number of members. Sequence $\{a_n\}$ has ① elements and, therefore, is complete, $\{b_n\}$ has $\frac{①}{2} - 1$, and $\{c_n\}$ has $2\frac{①}{3}$ elements. □

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 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]. We are not able to execute any infinite process physically but we assume the existence of such a process; moreover, 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 the observation. Indeed, we can observe only those members of a sequence for which there exist the corresponding numerals in the chosen numeral system; to better clarify this point the following example is discussed.

Example 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

$$\underbrace{1, 2}_{finite}, \quad \ldots \quad \underbrace{\frac{①}{2} - 2, \frac{①}{2} - 1, \frac{①}{2}, \frac{①}{2} + 1, \frac{①}{2} + 2}_{infinite}, \quad \ldots \quad \underbrace{① - 2, ① - 1, ①}_{infinite}. \tag{15}$$

[4] It is worthy to notice a deep relation of this observation to the Axiom of Choice. Since Theorem 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.

The first two numerals in (15) 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. □

In the light of the limitations concerning the process of sequential observations, 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 Let us consider the set A=$\{1,2,3,\ldots,2①\text{-}1,2①\}$ 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 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

$$\underbrace{1,2,\ 3,\ 4,\ \ldots\ ①\text{-}2,①\text{-}1,①}_{① \text{ steps}},①+1,①+2,①+3,\ldots,2①\text{-}1,2① \tag{16}$$

Analogously, if we start the process of the sequential counting from 5, the process arrives at maximum to $① + 4$:

$$1,2,3,4,\underbrace{5\ \ldots\ ①\text{-}1,①,①+1,①+2,①+3,①+4}_{① \text{ steps}},①+5,\ldots,2①\text{-}1,2① \tag{17}$$

The corresponding complete sequence used in this case is $\{n+4 : ①\}$. We can also change the length of the step in the counting sequence and consider, for instance, the complete sequence $\{2n-1 : ①\}$:

$$\underbrace{1,2,3,4,\ \ldots\ ①\text{-}1,①,①+1,①+2,\ \ldots\ 2①\text{-}3,2①\text{-}2,2①\text{-}1}_{① \text{ steps}},2① \tag{18}$$

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

The introduced definition of the sequence allows us to work not only with the first but with any element of any sequence if the element of our interest is expressible in the chosen numeral system independently whether the sequence under our study has a finite or an infinite number of elements. Let us use this new definition for studying infinite sets of numerals, in particular, for calculating the number of points at the

interval $[0,1)$ (see [31,33]). To do this we need a definition of the term 'point' and mathematical tools to indicate a point. 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 giving this definition we have not used the usual formulation "*x belongs to the set,* $\mathbb{R}$*, of real numbers*". This has been done because we can express coordinates only by numerals and different choices of numeral systems lead to different sets of numerals and, as a result, to different sets of numbers observable through the chosen numerals. In fact, we can express coordinates only after we have fixed a numeral system (our instrument of the observation) and this choice defines which points we can observe, namely, points having coordinates expressible by the chosen numerals. This situation is typical for natural sciences where it is well known that instruments influence the results of observations. Remind 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 the chosen numeral system and, as a result, we obtain the number of points at the interval $[0,1)$. Different variants (see [31,33]) 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

$$(0.a_1a_2\ldots a_{(①-1)}a_①)_b, \qquad a_i \in \{0,1,\ldots b-2,b-1\}, \quad 1 \le i \le ①. \tag{19}$$

Then, the number of numerals (19) gives us the number of points within the interval $[0,1)$ that can be expressed by these numerals. Note that a number using the positional numeral system (19) cannot have more than grossone digits (contrarily to sets discussed in Example 3) because a numeral having $g > ①$ digits would not be observable in a sequence. In this case ($g > ①$) such a record becomes useless in sequential computations because it does not allow one to identify numbers entirely since $g - ①$ numerals remain non observed.

Theorem 2 *If coordinates of points* $x \in [0,1)$ *are expressed by numerals (19), then the number of the points x over* $[0,1)$ *is equal to* $b^①$.

Proof. In the numerals (19) there is a sequence of digits, $a_1a_2\ldots a_{(①-1)}a_①$, used to express the fractional part of the number. Due to the definition of the sequence and Theorem 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 $\{0,1,\ldots,b-1\}$ that leads to $b^①$ possible combinations. Hence, the positional numeral system using the numerals of the form (19) can express $b^①$ numbers. □

Corollary 1 *The number of numerals*

$$(a_1a_2a_3\ldots a_{①-2}a_{①-1}a_①)_b, \qquad a_i \in \{0,1,\ldots b-2,b-1\}, \qquad 1 \le i \le ①, \tag{20}$$

expressing integers 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 2 and is so omitted. □

Corollary 2 *If coordinates of points* $x \in (0,1)$ *are expressed by numerals (19), then the number of the points x over* $(0,1)$ *is equal to* $b^{①}-1$.

Proof. The proof follows immediately from Theorem 2. □

Note that Corollary 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, expressed by the numeral $0.00\ldots0$ with ① zero digits after the decimal point) has been excluded from the first set in order to obtain the second one.

4 The Turing Machines observed through the lens of the Grossone Methodology

In this Section the different types of Turing machines introduced in Section 2 are analyzed and observed by using as instruments of the observation the Grossone language and methodology presented in Section 3. In particular, new results for Multi-tape Turing machines are presented and discussed.

Before starting the discussion, it is useful to recall the main results from the previous Section: (i) any infinite sequence can have maximum ① elements; (ii) the elements which we are able to observe in this sequence depend on the adopted numeral system.

Then, in order to be able to read and to understand the output of a Turing machine, writing its output on the tape using an alphabet Σ containing b symbols $\{0,1,\ldots b-2,b-1\}$ where b is a finite number, the researcher (the user) should know a positional numeral system $\mathcal{U}$ with an alphabet $\{0,1,\ldots u-2,u-1\}$ where $u \geq b$, otherwise the output cannot be decoded by the user. Moreover, the researcher must be able to observe a number of symbols at least equal to the maximal length of the output sequence that can be computed by machine, otherwise the user is not able to interpret the obtained result (see [42] for a detailed discussion).

In this Section, a first set of results aims to specify, with higher accuracy with respect to that provided by the mathematical language developed by Cantor and adopted by Turing, how and when the computations performed by a Multi-tape Turing machine can be observed in a sequence. Moreover, it is shown that the Grossone language and methodology will allow us to perform a more accurate investigation of situations interpreted traditionally like equivalences among different Multi-tape machines and among Multi and Single-tape machines.

4.1 Observing computations performed by a Multi-tape Turing machine

Before starting to analyze the computations performed by a k-tapes Turing machine (with $k \geq 2$) $\mathcal{M}_K = \left\langle Q, \Gamma, \overline{b}, \Sigma, q_0, F, \delta^{(k)} \right\rangle$ (see (1), Section 2.2), it is worth to make some considerations about the process of observation itself in the light of the Grossone methodology. As discussed above, if we want to observe the process of computation performed by a Turing machine while it executes an algorithm, then we have to execute 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 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.

Since our observations are made in a sequence, 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 in a way) 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 $[0,1)$, up to $b^{①}$ numerals of the type (19) can be used to identify moments of time but not more than grossone of them can be observed in a sequence. Moreover, 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. The structure of the language we use to describe the process imposes what we can say about the process (see [42] for a detailed discussion).

On the basis of the considerations made above, we should choose the accuracy (granularity) of the process of the observation of a Turing machine; for instance we can choose a single operation of the machine such as reading a symbol from the tape, or moving the tape, etc. However, 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 (see Section 2).

Moreover, concerning the output of the machine, we consider the symbols written on all the k tapes of the machine by using, on each tape i, with $1 \leq i \leq k$, the alphabet Σ_i of the tape, containing b_i symbols, plus the blank symbol ($\overline{b}$). Due to the definition of complete sequence (see Section 3) on each tape at least ① symbols can be produced and observed. This means that on a tape i, after the last symbols belonging to the tape alphabet Σ_i, if the sequence is not complete (i.e., if it has less than ① symbols) we can consider a number of blank symbols ($\overline{b}$) necessary to complete the sequence. We say that we are considering a *complete output* of a k-tapes Turing machine when on each tape of the machine we consider a complete sequence of symbols belonging to $\Sigma_i \cup \{\overline{b}\}$.

Theorem 3 *Let $\mathcal{M}_K = \left\langle Q, \Gamma, \overline{b}, \Sigma, q_0, F, \delta^{(k)} \right\rangle$ be a k-tapes, $k \geq 2$, Turing machine. Then, a complete output of the machine will results in k① symbols.*

Proof. Due to the definition of the complete sequence, on each tape at maximum ① symbols can be produced and observed and thus by considering a complete sequence on each of the k tapes of the machine the complete output of the machine will result in $k①$ symbols. □

Having proved that a complete output that can be produced by a *k*-tapes Turing machine results in $k①$ symbols, it is interesting to investigate what part of the complete output produced by the machine can be observed in a sequence taking into account that it is not possible to observe in a sequence more than ① symbols (see Section 3). As examples, we can decide to make in a sequence one of the following observations: (i) ① symbols on one among the *k*-tapes of the machine, (ii) $\frac{①}{k}$ symbols on each of the *k*-tapes of the machine; (iii) $\frac{①}{2}$ symbols on 2 among the *k*-tapes of the machine, an so on.

Theorem 4 *Let* $\mathcal{M}_K = \left\langle Q, \Gamma, \bar{b}, \Sigma, q_0, F, \delta^{(k)} \right\rangle$ *be a k-tapes,* $k \geq 2$*, Turing machine. Let M be the number of all possible complete outputs that can be produced by* $\mathcal{M}_K$*. Then it follows* $M = \prod_{i=1}^{k} (b_i + 1)^{①}$.

Proof. Due to the definition of the complete sequence, on each tape *i*, with $1 \leq i \leq k$, at maximum ① symbols can be produced and observed by using the b_i symbols of the alphabet Σ_i of the tape plus the blank symbol ($\bar{b}$); as a consequence, the number of all the possible complete sequences that can be produced and observed on a tape *i* is $(b_i + 1)^{①}$. A complete output of the machine is obtained by considering a complete sequence on each of the the *k*-tapes of the machine, thus by considering all the possible complete sequences that can be produced and observed on each of the k tapes of the machine, the number *M* of all the possible complete outputs will results in $\prod_{i=1}^{k} (b_i + 1)^{①}$. □

As the number $M = \prod_{i=1}^{k} (b_i + 1)^{①}$ of complete outputs that can be produced by $\mathcal{M}_K$ is larger than grossone, then there can be different sequential enumerating processes that enumerate complete outputs in different ways, in any case, each of these enumerating sequential processes cannot contain more than grossone members (see Section 3).

4.2 Equivalences among different Multi-tape machines and among Multi and Single-tape machines

In the classical framework *k*-tape Turing machines have the same computational power of Single-tape Turing machines and given a Multi-tape Turing Machine $\mathcal{M}_K$ it is always possible to define a Single-tape Turing Machine which is able to fully simulate its behavior and therefore to completely execute its computations. As showed for Single-tape Turing machine (see [42]), the Grossone methodology allows us to give a more accurate definition of the equivalence among different machines as it provides the possibility not only to separate different classes of infinite sets with respect to their cardinalities but also to measure the number of elements of some of them. With reference to Multi-tape Turing machines, the Single-tape Turing Machines adopted

for their simulation use a particular kind of tape which is divided into tracks (multi-track tape). In this way, if the tape has m tracks, the head is able to access (for reading and/or writing) all the m characters on the tracks during a single operation. This tape organization leads to a straightforward definition of the behavior of a Single-tape Turing machine able to completely execute the computations of a given Multi-tape Turing machine (see Section 2.2). However, the so defined Single-tape Turing machine $\mathcal{M}$, to simulate t computational steps of $\mathcal{M}_K$, needs to execute $O(t^2)$ transitions (t^2+t in the worst case) and to use an alphabet with $2^k(|\Sigma_1|+1)\prod_{i=2}^{k}(|\Sigma_i|+2)$ symbols (again see Section 2.2). By exploiting the Grossone methodology is is possibile to obtain the following result that has a higher accuracy with respect to that provided by the traditional framework.

Theorem 5 *Let us consider* $\mathcal{M}_K = \left\langle Q, \Gamma, \bar{b}, \Sigma, q_0, F, \delta^{(k)} \right\rangle$*,a k-tapes,* $k \geq 2$*, Turing machine, where* $\Sigma = \bigcup_{i=1}^{k} \Sigma_i$ *is given by the union of the symbols in the k tape alphabets* $\Sigma_1, \ldots, \Sigma_k$ *and* $\Gamma = \Sigma \cup \{\bar{b}\}$*. If this machine performs t computational steps such that*

$$t \leqslant \frac{1}{2}(\sqrt{4①+1}-1), \tag{21}$$

then there exists $\mathcal{M}' = \{Q', \Gamma', \bar{b}, \Sigma', q_0', F', \delta'\}$*, an equivalent Single-tape Turing machine with* $|\Gamma'| = 2^k(|\Sigma_1|+1)\prod_{i=2}^{k}(|\Sigma_i|+2)$*, which is able to simulate* $\mathcal{M}_K$ *and can be observed in a sequence.*

Proof. Let us recall that the definition of $\mathcal{M}'$ requires for a Single-tape to be divided into $2k$ tracks; k tracks for storing the characters in the k tapes of $\mathcal{M}_K$ and k tracks for signing through the marker $\downarrow$ the positions of the k heads on the k tapes of $\mathcal{M}_k$ (see Section 2.2). The transition function $\delta^{(k)}$ of the k-tapes machine is given by $\delta^{(k)}(q_1, a_{i1}, \ldots, a_{ik}) = (q_j, a_{j_1}, \ldots, a_{j_k}, z_{j_1}, \ldots, z_{j_k})$, with $z_{j_1}, \ldots, z_{j_k} \in \{R, L, N\}$; as a consequence the corresponding transition function δ' of the Single-tape machine, for each transition specified by $\delta^{(k)}$ must individuate the current state and the position of the marker for each track and then write on the tracks the required symbols, move the markers and go in another internal state. For each computational step of $\mathcal{M}_K$, $\mathcal{M}'$ must execute a sequence of steps for covering the portion of tapes between the two most distant markers. As in each computational step a marker can move at most of one cell and then two markers can move away each other at most of two cells, after t steps of $\mathcal{M}_K$ the markers can be at most $2t$ cells distant, thus if $\mathcal{M}_K$ executes t steps, $\mathcal{M}'$ executes at most: $2\sum_{i=1}^{t} i = t^2+t$ steps. In order to be observable in a sequence the number t^2+t of steps, performed by $\mathcal{M}'$ to simulate t steps of $\mathcal{M}_K$, must be less than or equal to ①. Namely, it should be $t^2+t \leqslant ①$. The fact that this inequality is satisfied for $t \leqslant \frac{1}{2}(\sqrt{4①+1}-1)$ completes the proof. □

5 Concluding Remarks

In the paper, Single and Multi-tape Turing machines have been described and observed through the lens of the Grossone language and methodology. This new language, differently from the traditional one, makes it possible to distinguish among infinite sequences of different length so enabling a more accurate description of Single

and Multi-tape Turing machines. The possibility to express the length of an infinite sequence explicitly gives the possibility to establish more accurate results regarding the equivalence of machines in comparison with the observations that can be done by using the traditional language.

It is worth noting that the traditional results and those presented in the paper do not contradict one another. They are just written by using different mathematical languages having different accuracies. Both mathematical languages observe and describe the same objects – Turing machines – but with different accuracies. As a result, both traditional and new results are correct with respect to the mathematical languages used to express them and correspond to different accuracies of the observation. This fact is one of the manifestations of the relativity of mathematical results formulated by using different mathematical languages 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 unique when viewed by a weaker lens.

Specifically, the Grossone language has allowed us to give the definition of *complete output* of a Turing machine, to establish when and how the output of a machine can be observed, and to establish a more accurate relationship between a Multi-tape Turing machine and a Single-tape one which simulates its computations. Future research efforts will be geared to apply the Grossone language and methodology to the description and observation of new and emerging computational paradigms.

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. V. Benci and M. Di Nasso. Numerosities of labeled sets: a new way of counting. *Advances in Mathematics*, 173:50–67, 2003.
4. G. Cantor. *Contributions to the founding of the theory of transfinite numbers*. Dover Publications, New York, 1955.
5. A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.
6. J.H. Conway and R.K. Guy. *The Book of Numbers*. Springer-Verlag, New York, 1996.
7. S. Barry Cooper. *Computability Theory*. Chapman Hall/CRC, 2003.
8. S. De Cosmis and R. De Leone. The use of grossone in mathematical programming and operations research. *Applied Mathematics and Computation*, 218(16):8029–8038, 2012.
9. L. D'Alotto. Cellular automata using infinite computations. *Applied Mathematics and Computation*, 218(16):8077–8082, 2012.
10. M. Davis. *Computability & Unsolvability*. Dover Publications, New York, 1985.
11. K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.
12. P. Gordon. Numerical cognition without words: Evidence from Amazonia. *Science*, 306(15 October):496–499, 2004.
13. J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading Mass., 1st edition, 1979.
14. D.I. Iudin, Ya.D. Sergeyev, and M. Hayakawa. Interpretation of percolation in terms of infinity computations. *Applied Mathematics and Computation*, 218(16):8099–8111, 2012.
15. S.C. Kleene. *Introduction to metamathematics*. D. Van Nostrand, New York, 1952.
16. A.N. Kolmogorov. On the concept of algorithm. *Uspekhi Mat. Nauk*, 8(4):175–176, 1953.
17. A.N. Kolmogorov and V.A. Uspensky. On the definition of algorithm. *Uspekhi Mat. Nauk*, 13(4):3–28, 1958.

18. G.W. Leibniz and J.M. Child. *The Early Mathematical Manuscripts of Leibniz*. Dover Publications, New York, 2005.
19. T. Levi-Civita. Sui numeri transfiniti. *Rend. Acc. Lincei, Series 5a*, 113:7–91, 1898.
20. G. Lolli. Infinitesimals and infinites in the history of mathematics: A brief survey. *Applied Mathematics and Computation*, 218(16):7979–7988, 2012.
21. M. Margenstern. Using grossone to count the number of elements of infinite sets and the connection with bijections. *p-Adic Numbers, Ultrametric Analysis and Applications*, 3(3):196–204, 2011.
22. M. Margenstern. An application of grossone to the study of a family of tilings of the hyperbolic plane. *Applied Mathematics and Computation*, 218(16):8005–8018, 2012.
23. A.A. Markov Jr. and N.M. Nagorny. *Theory of Algorithms*. FAZIS, Moscow, second edition, 1996.
24. J.P. Mayberry. *The Foundations of Mathematics in the Theory of Sets*. Cambridge University Press, Cambridge, 2001.
25. I. Newton. *Method of Fluxions*. 1671.
26. M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2000.
27. 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.
28. E. Post. Finite combinatory processes – formulation 1. *Journal of Symbolic Logic*, 1:103–105, 1936.
29. A. Robinson. *Non-standard Analysis*. Princeton Univ. Press, Princeton, 1996.
30. E.E. Rosinger. Microscopes and telescopes for theoretical physics: How rich locally and large globally is the geometric straight line? *Prespacetime Journal*, 2(4):601–624, 2011.
31. Ya.D. Sergeyev. *Arithmetic of Infinity*. Edizioni Orizzonti Meridionali, CS, 2003.
32. Ya.D. Sergeyev. Blinking fractals and their quantitative analysis using infinite and infinitesimal numbers. *Chaos, Solitons & Fractals*, 33(1):50–75, 2007.
33. Ya.D. Sergeyev. A new applied approach for executing computations with infinite and infinitesimal quantities. *Informatica*, 19(4):567–596, 2008.
34. 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.
35. Ya.D. Sergeyev. Numerical computations and mathematical modelling with infinite and infinitesimal numbers. *Journal of Applied Mathematics and Computing*, 29:177–195, 2009.
36. 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.
37. Ya.D. Sergeyev. Counting systems and the First Hilbert problem. *Nonlinear Analysis Series A: Theory, Methods & Applications*, 72(3-4):1701–1708, 2010.
38. Ya.D. Sergeyev. Lagrange Lecture: Methodology of numerical computations with infinities and infinitesimals. *Rendiconti del Seminario Matematico dell'Università e del Politecnico di Torino*, 68(2):95–113, 2010.
39. Ya.D. Sergeyev. Higher order numerical differentiation on the infinity computer. *Optimization Letters*, 5(4):575–585, 2011.
40. Ya.D. Sergeyev. On accuracy of mathematical languages used to deal with the Riemann zeta function and the Dirichlet eta function. *p-Adic Numbers, Ultrametric Analysis and Applications*, 3(2):129–148, 2011.
41. Ya.D. Sergeyev. Using blinking fractals for mathematical modelling of processes of growth in biological systems. *Informatica*, 22(4):559–576, 2011.
42. Ya.D. Sergeyev and A. Garro. Observability of Turing machines: A refinement of the theory of computation. *Informatica*, 21(3):425–454, 2010.
43. A.M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of London Mathematical Society, series 2*, 42:230–265, 1936-1937.
44. M.C. Vita, S. De Bartolo, C. Fallico, and M. Veltri. Usage of infinitesimals in the Menger's Sponge model of porosity. *Applied Mathematics and Computation*, 218(16):8187–8196, 2012.
45. A. Žilinskas and J. Žilinskas. Interval arithmetic based optimization in nonlinear regression. *Informatica*, 21(1):149–158, 2010.
46. J. Wallis. *Arithmetica infinitorum*. 1656.
47. G.W. Walster. *Compiler Support of Interval Arithmetic With Inline Code Generation and Nonstop Exception Handling*. Tech. Report, Sun Microsystems, 2000.
48. A.A. Zhigljavsky. Computing sums of conditionally convergent and divergent series using the concept of grossone. *Applied Mathematics and Computation*, 218(16):8064–8076, 2012.
49. A. Žilinskas. On strong homogeneity of two global optimization algorithms based on statistical models of multimodal objective functions. *Applied Mathematics and Computation*, 218(16):8131–8136, 2012.