NON-CLASSICAL TURING MACHINES: EXTENDING THE NOTION OF COMPUTATION

Jiří Wiedermann Jan van Leeuwen

Institute of Computer Science of AS CR, Prague, Czech Republic
jiri.wiedermann@cs.cas.cz

Dept. of Information and Computing Science, Utrecht University, the Netherlands
J.vanLeeuwen1@uu.nl

Abstract

The goal of this paper is to broaden our understanding of computations based on the classical paradigm of Turing machines. To this end we will point to models of non-classical Turing machines that capture computational behavior of contemporary or emerging computing technologies and that have been designed and studied by the authors. The guiding design principles of such machines have been the currently existing or foreseeable technologies such as interactivity, non-uniform evolution, unbounded time behavior, process cooperation, and relativistic time-space effects. The computational behavior of these models and the fact that they mirror key aspects of present or envisaged technologies jointly offer compelling reasons for modifying the traditional and persisting view of computation. This change is quite a profound one: computations should no longer be seen as finite processes whose parameters are fixed before the start of processing. Rather, computations are potentially infinite evolutionary processes whose parameters can change during the processing in an unpredictable way in interaction with their environment.

1. Introduction

Classical Turing machines have established themselves as 'etalons' of effective machine computations. While weaker forms of computation have natural models and are studied since the time of finite automata, until recently the study of stronger forms of computation "beyond Turing" has been driven more by mathematical curiosity for trespassing the computational limits of classical Turing machines (cf. [6]) than by the potential of current scientific and engineering knowledge. The situation has changed with our deeper understanding of natural and artificial processes that, intuitively, should also be classified as computations despite the fact that they can produce a computational behavior that is not captured by the classical Turing

The research of the first author was partially supported by RVO 67985807 and GA ČR grant No. 15-04960S

machine paradigm. How can we model such computations? This paper will describe selected non-classical Turing machine models designed and studied by the authors of this paper in recent years. The guiding design principle of all these models has been the potential of the existing or foreseeable technologies that allow crossing the computational boundaries of classical Turing machines by exploiting interactivity, non-uniform evolution, unbounded time behavior, process cooperation, and relativistic time-space effects. Perhaps surprisingly, we show that the first ideas for such models can be traced back to Turing's famous 1936 paper, notably to his model of non-terminating circular a-machines [5]. These machines turn out to be equivalent to our so-called red-green Turing machines [12] modeling multi-process computational systems that are always up and running [11]. We also mention interactive Turing machines with advice as a model of the Internet [7] and relativistic Turing machines [15] inspired by certain theories from relativistic space-time physics. The computational power of all these models can be characterized in terms of levels of arithmetical hierarchy. It turns out that the computational power of reasonable variants of these machines reaches up to the second level of this hierarchy (class Σ_2), substantially beyond the power of classical machines. This result ranks the respective machines models among the class of so-called hypercomputers. What is perhaps more important and what we would like to stress by this paper, is the fact that our models capture the essential feature of existing or foreseeable data processing technologies and therefore extend, in a natural and compelling way, the prevailing traditional notion of computation: computation is what these non-classical models of Turing machines do.

Since the aim of the paper is to call attention to the evidence that the everyday practice of computation is well beyond the way described by still persisting notion of what is computation (namely, "what classical Turing machines do"), we restrict our attention to informal descriptions of selected models of non-classical Turing machines which are, in a sense, realistic. While doing so we point to their "non-classical" additional computational resources that on one hand increase their computational power beyond those of the classical Turing machines and, on the other hand, indicate how, or to what extent these extra features correspond to computational properties of contemporary computing technologies. More details about the non-classical models at hand can be found in the original papers.

The structure of the paper is as follows. In Section 2 we present our basic model — interactive Turing machines with advice — capturing three aspects of contemporary computing: interactivity, non-uniform evolution, and unbounded time behavior. In Section 3 we show that the previous three aspects of contemporary computing as exemplified by the ITM/A's are sufficient for simulating the Internat. An aspect omitted in the previous modeling of contemporary computing systems — namely time-unbounded computations seen as an interplay of communicating processes — is investigated in Section 4 with the help of so-called red-green Turing machines. Their surprising connection to Turing's non-terminating circular a-machines, described in the original Turing's 1936 paper, is discussed in Section 5. An, in a sense, futuristic model of a Turing machine — the so-called relativistic Turing machine — that works under so-far not experimentally verified assumptions of relativistic black-hole physics, is considered in section 6. Finally, in Section 7 we summarize the arguments against sustaining the presently still prevailing classical notion of computations. Section 8 contains the conclusions.

To close this introductory part we remark that "non-classical" models of Turing machines

have been proposed and studied by many other authors. We refer to the existing literature on hypercomputation for it. In many cases, almost as a rule, such models have been developed with the aim to obtain machines provably computationally stronger than classical Turing machines, with no attention paid to perspectives on their practical realizability. The first paper along this lines has been Turing's 1939 paper [6], introducing Turing machines with oracles enabling them to solve classically non-computable problems. This line of thinking has been followed by many authors, especially at the end of the twentieth century, without noticing that already in the present day (even at that time) information technologies did have a potential to trump the classical models. Argumentation in favor of the last claim will be the content of this paper.

2. Interactive Turing machines with advice

We start our excursion into non-classical models of Turing machine by introducing the basic and most versatile model of such machines capturing three ingredients of contemporary computing: interactivity, non-uniform evolution, and unbounded time behavior. It is the so-called *interactive Turing machines with advice* (ITM/A). At first sight, this model does not look like a model of an existing piece of computational technology. Fortunately, this appearance is deceptive: in the next section we show that this model is perfectly capable to simulate (a simplified version of) the Internet. It is its simplicity and mathematical elegance that designates this model to be the reference model among the models of non-classical Turing machine aspiring to capture the potential of current computing technologies. The model extends the well-known and well-studied model of (ordinary) Turing machines with advice in computational complexity theory (cf. [4]).

The ITM/A is a Turing machine whose architecture is changed in two ways:

- instead of an input and output tape, the interactive Turing machine has an input port and an output port allowing the reading or writing of potentially infinite streams of (possibly empty) symbols
- the machine is enhanced by a special, so-called advice tape that, upon a request, allows insertion of a possibly non-computable external information (called *advice*) into the computation. The advice takes the form of a finite string of symbols that becomes available to the machine. This string must not depend on the concrete stream of input symbols read by the machine until that time; it can only depend on the number of those symbols. The length of of the advice string is called *advice size*.

Note that an ITM/A works as a transducer — it translates infinite streams of input symbols into infinite streams of output symbols. This mechanism provides *interactiveness* to the model — the machine can communicate with the environment, reflect its changes, get feedback, etc. Also note that the advice is different from an oracle as also considered in computability theory: an oracle value can depend on the current input (cf. [4]), whereas an advice value must not. The advice may contain, e.g., a description of a new Turing machine. An ITM/A with such an advice can, after calling its advice, behave differently than before. This enables one to capture potentially time-unbounded *evolution* — machine development over generations of its

successors. This development can be *non-uniform* since advice can provide the machine at hand with completely non-computable information. We conclude that the ITM/A represents a non-uniform model of interactive, evolving, and time-unbounded computation. In the next section we prove that these three properties are sufficient for simulating the Internet.

The mechanism of advice functions is very powerful and can provide an ITM/A with arbitrary non-computable "assistance". For theoretical and practical reasons it is useful to restrict the size of advice growth in ITM/A's to polynomial functions. (With advice functions that grow exponentially one could encode arbitrary oracles in advice.)

As an example of the power of advice we sketch a proof of the following theorem:

Theorem 2.1 *There is an ITM/A $\mathcal{A}$ which for any given classical TM $\mathcal{M}$ and its input w determines in finitely many steps whether $\mathcal{M}$ will halt on input w or not.*

Sketch of the proof: Let $\mathcal{N}_n$ be the classical Turing machine of size n_1 whose running time is maximal over all inputs w' of size n_2 for any $n_1 > 0$ and $n_2 > 0$ such that $n_1 + n_2 = n$. Define an advice function of $\mathcal{A}$ that for each n returns description $< N_n >$ of $\mathcal{N}_n$ and input w'. Note that the value of this advice function depends only on the input size n.

Let the size of $\mathcal{M}$'s description $< M >$ plus the size of w be n. Then $\mathcal{A}$ on input $(< M >, w)$ of length n calls its advice and gets the description of $\mathcal{N}_n$ and string w'. Next $\mathcal{A}$ simulates one step of M on w and one step of $\mathcal{N}_n$ on w'. Clearly, M on w halts if and only if its simulation by $\mathcal{A}$ ends not later than simulation of $\mathcal{N}_n$ on w'.

(Note that, in this simulation, interactivity of $\mathcal{A}$ has not been used. This is because the goal here is merely to show the principle of constructing an advice serving our purpose.)

□

The ITM/A with the "halting advice" from the proof of the above theorem will be denoted as ITM/H. It is almost obvious that an ITM/H recognizes languages from Σ_2 and accepts languages from Δ_2.

3. Simulating the Internet by an ITM/A

In our quest for Turing-machine-like models of contemporary information processing technologies, let us start with the most widely spread computational technology used today — the Internet. Can equivalent computations be performed by classical Turing machines? Of course, the answer is negative, for at least three reasons. First, the Internet performs interactive computations. It takes infinite streams of inputs (not known before the start of processing) and produces infinite streams of outputs. Second, the Internet evolves while computing, and it evolves in two different ways: either its software, or its hardware is being updated at some sites. Moreover, new sites are being added to or deleted from the network. Third, the evolution of the Internet is not computable — it evolves non-uniformly, in an unpredictable manner

depending on external factors. Obviously, a classical Turing machine cannot process infinite input streams and its transition function cannot change during a computation. But the lastly mentioned obstacles point to the way how a classical Turing machine must be modified in order to be able to simulate the Internet. In the previous section we have seen such a model, viz the ITM/A.

Simulation of the Internet by an ITM/A can only be done under some simplifying assumptions concerning its operation. This is because the Internet is represented by a real piece of technology that obeys many physical and technological restrictions. On the other hand, an ITM/A is a highly abstract device which, when compared to the Internet, is ridiculously simple. To this end, we will consider the following *simplified Internet*: for simplicity assume that all sites in the Internet work synchronously and that each site has a unique internet address — a natural number. Each site can be modeled by an interactive Turing machine without advice. The sites have available a table of addresses of all sites in the network. Based on these addresses, sites can exchange messages. A message consists of a single symbol and contains the address of the sender. A message arrives at its destination in unit time. When more messages are sent to the same site they queue at that site and are processed sequentially, one message at a time.

When thinking about a simulation of this simplified version of the Internet by an ITM/A we will take advantage of two facts: (i) at each time, the Internet consists of a finite number of sites, each of which can be simulated by a Turing machine and (ii) the potentially infinite running time of the Internet can be split into finite intervals during which the Internet is "stable" — no site updates of its hardware of software occur during these intervals and the number of sites does not change.

Next we will describe an ITM/A $\mathcal{I}$ simulating the simplified version of the Internet. Prior to do so we must make an other assumption enabling such a simulation. Namely, our model of the Internet still assumes extra inputs to and extra outputs from each site, whereas our ITM/A has but one input and output port. In order to deal with this constraint, we sequentialize the inputs to the Internet sites as follows. At any time t, we number each input by the address of the site to which the input at hand belongs. For each time t, the set of all inputs to all sites sorted w.r.t. site addresses forms a sequential batch and this batch enters sequentially the input port of the simulating machine. Analogously, the outputs from each site are labeled by its address and sent to the output port of the simulating machine.

Let t_i be the i-th interval of the Internet's stability and let during this time interval $A_{t_i} \subsetneq \{1, 2, 3, \dots\}$ be the finite set of site addresses. The set of addresses is kept and managed by machine $\mathcal{I}$. Machine $\mathcal{I}$ keeps set A_{t_i} ordered according to their addresses. To each site with address $j \in A_{t_i}$ there corresponds a specific Turing machine S_j with a special input and output port simulating computations of that site. We assume that S_js are universal Turing machines. At any time t, each S_j has a complete description of that site, i.e., the program and the instantaneous description of S_j at time t. Based on this, $\mathcal{I}$ can perform, in a sequential manner, one computational step at each site. This step consist of reading a symbol from the input port, reading a received message, performing one computational step, and possibly sending a message to other sites. Doing so for each site, $\mathcal{I}$ performs one computational step of the Internet. In this manner $\mathcal{I}$ proceeds until it reaches the end of the stability interval t_i.

In order to perform the first step in the stable interval t_{i+1}, new descriptions of those sites that have changed or that have been added to or deleted from the Internet must be obtained. As the first thing, $\mathcal{I}$ calls its advice and in return it gets the set $A_{t_{i+1}}$ of new site addresses. Then, for each $j \in A_{t_{i+1}}$ machine $\mathcal{I}$ calls advice again, now with parameter j, and as a return it gets the description of machine S_j's new program and new instantaneous description. Of course, some S_j's can continue their processing without any "software" change, some machines can emerge for the first time or some can vanish. The latter case is recognized by $\mathcal{I}$ when advice for some $j \in A_{t_{i+1}}$ returns an empty string. Then $\mathcal{I}$ resumes to its simulation similarly as in interval t_i.

Of course, the sketched simulation is far from optimal (e.g., the management of the address table —- allocation of new sites and their removal can be done deterministically as in the case of the real Internet). On the other hand, the simulation can be made more realistic, e.g., for each message the advice can return the delivery delay of a message sent from site j to site k. For a more detailed description of the simulation see the original paper [7].

Note that in our simulation the advice is solely used for guessing the future development of the Internet — the non-predictable actions of its users, or the non-predictable fluctuations of the message delays, etc. The reverse simulation of any ITM/A by the Internet is not possible, due to the fact that the Internet has no means to guess advice functions recording the Internet's future behavior.

As far as the complexity of the simulation is concerned, not that at each step $\mathcal{I}$ updates the description of the entire Internet. In this estimation the complexity of site description via advice calls is already included. From the previous description it is clear that the simulation by $\mathcal{I}$ can be organized in such a way that the simulation of one step of a simplified Internet at time t by $\mathcal{I}$ can be done in time proportional to the descriptive complexity of the simplified Internet at time t.

Theorem 3.1 *There is an ITM/A that can simulate one step of a simplified Internet in linear time w.r.t. the size of the simplified Internet at that time.*

4. Red-green Turing machines

In the previous section we presented interactive Turing machine with advice which captured three distinguishing features of contemporary information processing systems: interactivity, non-uniform evolution and potentially unbounded operation. In doing so we have abstracted from another important property of current computational systems. Namely, as a rule such systems are multiprocessor systems alternately running many processes whose number usually vastly surpasses the number of available processors. In such systems control goes from process to process and, whenever a process finishes its task, it executes an instruction that explicitly transfers control to an other process. Simplifying and abstracting this view further, the work of such multi-process systems can be modeled as an interaction between red and green processes, where occasionally the control goes from red to green processes, or vice versa, as dictated by the processes themselves. Can this property of current computational systems be also captured

by a model based on a Turing machine? And what is the computational power of such systems?

The answer to these question has been given in paper [11]. Here, so-called *red-green Turing machines* (rgTMs) have been designed and investigated as a formalized model of contemporary multi-processor systems that are always on and running. The model is built on top of a classical Turing machine whose acceptance criterion is changed as follows. The set of states of the underlying machine is split into two disjoint sets: the set of green states, and the set of red states, respectively. There are no halting states. A computation of a rgTM proceeds as in the classical case, changing between red and green states in accordance with the transition function. The moment of a state color change is called a *mind change*. A word of a formal language is said to be recognized if and only if the machine's computation after some time does not change it mind and "stabilizes" in green states. That is, from a certain time on, the machines keeps entering only green states. Otherwise, the input word gets rejected. In [11] we have investigated various aspects of red-green computations from the computability point of view. E.g., we showed that the computational power of rgTM increases with the number of mind changes allowed (their computational power increases along the so-called Ershov hierarchy, cf. [2]). By direct mutual simulations between rgTM's with a finite number of mind changes and ITM/H's one can prove that rgTM's recognize languages in Σ_2 and accept languages from Δ_2. This proves equivalence of rgTM's with ITM/H's. In fact, computations of red-green Turing machines exactly characterize the latter two classes.

A non-deterministic model of rgTM's can be defined as well. Interestingly, it appears that w.r.t. the number of mind changes, the nondeterministic model is demonstrably more powerful than the deterministic one [11].

The models similar to rgTM's have a rich history — cf. [11] for an overview. This suggests that, thanks to their simplicity and mathematical elegance, the rgTM's can serve as a bridging model among the various alternative models of potentially infinite computations.

5. Turing′s non-terminating circular a-machines

The non-terminating circular Turing machine (ntcTM) has been defined in the Turing's original 1936 paper [5]. Informally, it is a Turing machine that never halts and produces but a finite number of output symbols. Turing himself did not pay much attention to these machines — he has mentioned them as a logical alternative to (for him) much more interesting, so-called circle-free machines that never halt while producing an infinite number of output symbols. He had a good reason for preferring the latter machines — circle-free machines were designed for the task of computing infinite expansions of real numbers. For ntcTM's such good reason was not found and therefore they have remained almost unnoticed among various models of Turing machines.

Nevertheless we observed recently [12] that rgTM's can, quite surprisingly, be seen as a modern variant of ntcTM's. We sketch the connection between the two models. To this end we will view a *non-deterministic circular Turing machine* as a machine with separate input and output tapes.

On a given input, such machine prints a finite number of output symbols (not necessarily into different cells), and then either halts, or goes forever performing an infinite number of steps in which no output symbol is printed anymore. The relation between the computations of rgTM's and ntcTM's reveals the following theorem.

Theorem 5.1 *Let $\mathcal{A}$ be a ntcTM, let $\mathcal{R}$ be a rgTM, let x be an input string and let $f : \{0,1\} \to \{0,1\}^*$ be a partial function. Then $f(x)$ is computed by $\mathcal{A}$ if and only if $f(x)$ is computed by $\mathcal{R}$ with $2f(x)$ mind changes.*

Sketch of the proof: We describe the main ideas of the proof from [12].

If f is computed by $\mathcal{A}$, then $\mathcal{R}$ simulates non-output instructions of $\mathcal{A}$ step by step in green states. When $\mathcal{A}$ outputs a symbol then $\mathcal{R}$ outputs the same symbol and performs two mind changes (i.e., it performs state change *green* $\to$ *red* $\to$*green*). If $\mathcal{A}$ has no move from some configuration, then $\mathcal{R}$ switches to a red state and starts cycling in that state.

If $\mathcal{A}$ prints a finite number $f(x)$ of symbols without termination, then $\mathcal{R}$ accepts x in $2f(x)$ mind changes and stabilizes in a green state. If $\mathcal{A}$ keeps on producing an infinite number of output symbols, then $f(x)$ is undefined and obviously, $\mathcal{R}$ keeps changing its mind infinitely often as well.

Obviously, $\mathcal{R}$ computes f and, if $f(x)$ is a finite number, then $\mathcal{R}$ computes $f(x)$ in $2f(x)$ mind changes.

Conversely, let $\mathcal{R}$ be a machine accepting x with $2f(x)$ mind changes. $\mathcal{R}$ keeps a counter of its mind changes and at the occasion of every second mind change it increases this counter by one and prints it on its output tape, rewriting whatever has been written there (note that we have assumed that output symbols can be rewritten). Thus, after $2f(x)$ mind changes the output from $\mathcal{R}$ represents the value of $f(x)$. If x is not accepted by $\mathcal{R}$, $\mathcal{R}$ oscillates forever between red and green states and thus $\mathcal{A}$ produces an infinite number of outputs, effectively rejecting the input x for which $f(x)$ is undefined.

□

Thanks to the previous theorem we know that the computational power of rgTM's and ntcTM's is the same. Is it not amazing that an authentic, "old-fashioned" model considered by Turing himself appears to have a super-Turing computational power? And that, in fact, ntcTM's designed in 1936, model unbounded computational processes that are potentially performed by current computing technologies?

6. Relativistic TM's

Until the present day, no infinite process can be observed in finite time. If this would be possible we could realize rgTM's, since we could observe stabilization of a given red-green computation in green or red states. Yet in relativistic physics there are theories admitting such phenomena

(cf. [3]). Namely, within relativistic theoretical black-hole physics, in the so-called Malament-Hogarth spacetime, physicists have shown the existence of two different trajectories, T_1 and T_2, respectively, coming out from the same point, having the following property. The journey along T_1 takes infinite time, whereas for a traveler along T_2 only finite time will elapse. This can be used for performing infinite computations in finite time as follows.

The applications of this phenomenon to computing could be as follows. Let $\mathcal{M}$ be a classical Turing machine performing a possibly non-halting computation. Then, we let $\mathcal{M}$ follow trajectory T_1 while a person waiting for the result of $\mathcal{M}$'s computation (called the observer) will follow trajectory T_2. In our Universe, rotating black holes are the candidates for a Malament-Hogarth spacetime. Trajectory T_1 will be the orbit around the black hole (outside the so-called *event horizon*) while T_2 will be the trajectory of the observer falling towards the black hole. As long as this observer is outside of the event horizon the physical property of the black holes and that of their environment allows him to receive signals from the orbiting computer. The computer is programmed so that it sends a signal to the observer if and only if it halts. Thus, if a signal reaches the observer outside of the event horizon the the observe learns that the computation has stopped. However, as soon as the observer reaches the black hole inner horizon, for $\mathcal{M}$ infinite time will elapse. If no signal is obtained by the observer until that time, the computation of $\mathcal{M}$ did not halt in finite time. Unfortunately, obtaining/not obtaining a signal will be the last thing that an observer will observe. The observer cannot survive a fall into a black hole.

The existence of the Malament-Hogarth spacetime follows from the theory, but has not been confirmed empirically. Irrespective of whether Malament-Hogarth spacetime exists or not, in theory, as an *Gedankenexperiment*, one can design a so-called relativistic TM (a rTM) (cf. [15]) that exploits the above-mentioned effect. Basically, it is a classical TM enhanced by special, so-called signal states. There are three kinds of signal states: a query state, a YES-state, and a NO-state. Once a rTM enters a query state, the machine performs the following operation. It checks, in one step, whether a computation starting in a given configuration would halt or not. If it would halt, the machine enters a YES-state, otherwise it enters a NO-state. Based in this information the computation can proceed classically until next query state is entered or the computation halts.

In [15] we showed that relativistic computing has precisely the power of recognizing the Δ_2-sets of the Arithmetical Hierarchy, and that they can simulate rgTM's. If the underlying physical theory is accepted, this would lift the barrier of recursiveness to the Δ_2-level without violating any feasible thought experiments in General Relativity Theory.

7. Extending the notion of computation

We have presented several models of non-classical TM's whose computational power surpasses that of the classical TM's. What makes these models "non-classical" is their use of additional computational resources. In their most general form these resources include interactivity, non-uniform evolution, and potentially endless computation time. In a straightforward way these

resources are modeled by ITM/A's. We have shown in Section 3 that an ITM/A is capable to model (a simplified version of) the Internet and it is obvious that without any of the three above mentioned resources this would not be possible. In other words, a classical Turing machine cannot simulate the Internet. Or, we can put it as follows: a classical notion of computation is too weak to capture interactivity, non-uniform evolution, and potentially endless computation. But is this a good enough reason to abandon the classical notion of computation? Are there important reasons for changing the notion of computation based on the paradigm of classical Turing computations so as to capture the power of current computational technologies?

We believe that the answer should be positive, and should go as follows. The notion of computations based on classical Turing machines is very restrictive. It sees a computation as a finite process whose parameters are fixed during the entire processing, given *a priori*, before the start of the computation. This is to be compared with the modern view of computation which sees it as a potentially evolutionary process whose parameters can change during the processing in an unpredictable way in interaction with their environment. This view is supported by models that mirror the computational potential of existing technology. Therefore, it is no longer sustainable to dwell on a notion of computation that is based on the 80+ years old model of computing, no matter how basic and important it has been (and still is). For instance, the classical view of computation does not support interaction, learning, and evolution. It does not offer a framework for thinking about cognition or evolution over generations which are important constituents of computations that, as we believe, are going on in living organisms. The new, extended view of computations is not only quantitatively different from the classical view (finite vs. infinite processes), it is also qualitatively different (interactivity, non-uniform evolution). These are fundamental differences that cannot be neglected.

As a matter of fact, both classical and extended notion of computations as considered above are machine-oriented notions — they both relate to machine models of computations. However, with our recent deeper understanding of natural and artificial processes, we see that computation is a process that can be realized in many ways, not inevitably by (digital) machines. In this sense the notions of computation based on machine models of computing are still unnecessarily restrictive. With this idea in mind we have recently come with a new understanding of computation, the one that is based on what a computation does, rather than how it does what it does. Namely, what a computation does is knowledge generation, and the question, what mechanisms support this process, is only of secondary interest. This approach leads to a machine independent view of computation in which one can still speak about potentially endless interactive evolutionary knowledge generating processes. For more details about this approach, cf. [16].

8. Conclusions

The goal of presenting several particular models of non-classical Turing machines in this paper has been to support the thesis that on one hand these models are more powerful than classical Turing machines, and on the other hand, that they capture the main ingredients of contempo-

rary computing: interactivity, potential infinity of operation, and non-uniform evolution. The first fact alone is not sufficient to support a change of the classical notion of computation. However, the first and the second fact together give a strong support to the idea that time has come to extend the classical notion of computation based on computations of classical Turing machines to include also the computations as performed by modern computing devices in principle.

Describing these particular models of non-classical Turing machines in a sketchy way as we did, we have omitted a number of details and results related to these models. In the respective research there is still a lot of open space for further research. For instance, one can build on the models described in this paper and investigate their further properties and modifications. An interesting open problem along these lines is, e.g., the question of alternating rgTM's. How should such machines be defined? Can their computing power go beyond the class Σ_2? What about alternating rTM's? An other possibility is to consider various machines with restricted use of their advice. In paper [14] we showed that there is a variant of Turing machines that makes a restricted use of its advice, characterizing precisely the complements of recursive languages. An interesting non-classical model of finite automaton with a classical Turing machines in place of an oracle has been studied in [13]. The respective research was inspired by theoretical models for the human players and IBM's Watson supercomputer, respectively, playing the game of *Jeopardy!* [1]. Apparently, there are many examples of non-classical computations around us whose models are waiting to be discovered and studied.

References

[1] BBC NEWS, IBM's Watson supercomputer crowned Jeopardy king, February 17, 2011, http://www.bbc.co.uk/news/technology-12491688.

[2] Y.L. ERSHOV, A hierarchy of sets I, Algebra Logika 7 (1) (1968) 4774 (in Russian). Algebra Logic 7 (1) (1968) 2543 (English translation).

[3] G. ETESI, I. NÉMETI: Non-Turing computations via Malament-Hogarth spacetimes, *Int. J. Theor. Phys.* 41 (2002) 341-370, see also: http://arXiv.org/abs/gr-qc/0104023.

[4] R.M. KARP, R. LIPTON, Turing machines that take advice, *L'Enseignement Mathématique*, II^e Série, Tome XXVIII, 1982, pp. 191–209.

[5] A.M. TURING, On computable numbers, with an application to the Entscheidungsproblem, *Proc. London Math. Soc.* 42-2 (1936) 230-265; A correction, ibid., 43-2 (1937) 544-546.

[6] A.M. TURING, SysteM's of logic based on ordinals, *Proc. London Math. Soc.*, Series 2, 45 (1939) pp. 161-228.

[7] J. VAN LEEUWEN, J. WIEDERMANN, The Turing machine paradigm in contemporary computing, in: B. Engquist and W. Schmidt (Eds), *Mathematics Unlimited - 2001 and Beyond*, Springer-Verlag, 2001, pp. 1139-1155.

[8] J. VAN LEEUWEN, J. WIEDERMANN, Beyond the Turing limit: Evolving interactive systems. In: L. Pacholski and P. Ruzicka (Ed.), SOFSEM 2001: Theory and Practice of Informatics, Proc. 28th Conference, Lecture Notes in Computer Science Vol 2234, Springer-Verlag, Berlin, 2001, pp. 90-109.

[9] J. VAN LEEUWEN, J. WIEDERMANN, A Theory of Interactive Computation. Chapter in: D. Goldin, S.A. Smolka, and P. Wegner (Eds.), Interactive Computation: the New Paradigm, Springer-Verlag, Berlin, 2006, pp 119-142.

[10] J. VAN LEEUWEN, J. WIEDERMANN, How We Think of Computing Today. In: A. Beckmann, C. Dimitracopoulos, and B. Löwe (Eds.), Logic and Theory of Algorithms, 4th Conference on Computability in Europe (CiE 2008), Proceedings, Lecture Notes in Computer Science, Vol. 5028, Springer-Verlag, Berlin, 2008, pp. 579-593.

[11] J. VAN LEEUWEN, J. WIEDERMANN, Computation as an unbounded process. Theor. Computer Sci 429, pp. 202-212

[12] J. VAN LEEUWEN, J. WIEDERMANN, The computational power of Turing's non-terminating circular a-machines. In: S.B. Cooper & J. van Leeuwen, Alan Turing: His Work and Impact, Elsevier,2013, pp. 80-84.

[13] J. VAN LEEUWEN, J. WIEDERMANN, Question-answering and cognitive automata with background intelligence, Techn. Report UU-CS-2016-007, Department of Information and Computing Sciences, Utrecht University, May, 2016.

[14] J. VAN LEEUWEN, J. WIEDERMANN, Turing machines with one-sided advice and acceptance of the co-RE languages. Fundamenta Informaticae 153 (2017), pp. 347–366

[15] J. WIEDERMANN, J. VAN LEEUWEN, Relativistic computers and non-uniform complexity theory. Techn. Report 866 , Institute of Computer Science, Czech Academy of Sciences, Prague, 2002, also in: C.S. Calude, M.J. Dineen, and F. Peper (Eds), Unconventional models of computation, Proc 3rd Int Conference (UMC'2002), Lecture Notes in Computer Science Vol 2509, Springer- Verlag, Berlin, 2002, pp 287-299.

[16] J. WIEDERMANN, J. VAN LEEUWEN, What is Computation: An Epistemic Approach. (Invited talk). In Italiano, G.; Margaria-Steffen, T.; Pokorný, J.; Quisquater, J.J.; Wattenhofer, R. (ed.). SOFSEM 2015: Theory and Practice of Computer Science. LNCS 8939, Berlin : Springer, 2015, pp. 1-13