International Journal of Nanotechnology and Molecular Computation

January-March 2010, Vol. 2, No. 1

Table of Contents

Computational Model Based on Evolutionary Primitives:
Turing Machine Generalization

Oleg Granichin, Saint-Petersburg State University, Russia

Valentin Vasilev, Saint-Petersburg State University, Russia

ABSTRACT

This paper proposes generalization of Classical Turing machine scheme. New concept generalizes traditional concepts of "tape" and "tape cell". In particular, "tape cell" represents permanently running model of a dynamical system. "Natural" evolutions of cells proceed with jumpings. This model, for example, can describe systems with variable state spaces. Using this model one can avoid the problem of simulation continuous dynamic in the discrete time environment. Classical Turing machine is an extreme case of the proposed model.

Keywords: *Classical Computational Models, Hybrid Systems, Jumpings, Turing Machine*

INTRODUCTION

Turing machine (hereinafter, MT) is an abstract computing device, proposed in 1937 by mathematician Alan Turing (1936) as a mathematical model to describe algorithms. Initially, the concept of MT was being developed to answer the question whether for any mathematical statements it is possible to indicate the final sequence of instructions that could be carried out mechanically, one after another. In the same year A. Church (1936) proposed that any process which intuitively might be called a procedure could be implemented by a Turing machine. This hypothesis became known as the Church-Turing thesis.

Physical formulation of the Church-Turing principle: every finitely realizable physical system could be precisely modeled using the universal model of computer, operating by finite means. Practically, this set of finite resources is usually used as that of binary symbols (0, 1), and the fiinite realizability equivals to a finite number of cycles (the moments of movement of the head of the automaton).

However, physicists have emphasized the importance of quantum mechanics to computer science and the "exponential" advantage of quantum computers over classical. R. Feynman (1982) proved the impossibility of presenting the results of quantum mechanics in the classical universal computing device, even with the use of probabilistic algorithms. This is due to the fact that the number of elementary operations

DOI: 10.4018/jnmc.2010010103

of the computing device will grow exponentially with the number of degrees of freedom. Therefore, for accurate simulation of quantum systems Feynman proposed a theoretical model of a quantum computer.

D. Deutsch (1985) proposed to modify the Church-Turing thesis as follows: "Each course realizable physical system can be fully simulated modeling universal computing machine operating by finite means." D. Deutsch built a model of such a universal computer, based on the idea of quantum computing, and showed how a quantum computer can simulate physical systems, which are outside the area, affordable universal MT. Deutsch's model gives a new perspective on the problem of modern computing devices, but in essence as well as the MT remains discrete.

To date uses of this theoretical results prognoses the miniaturization of all the practically working computational devices. Once assumed that more powerful machines will require more space for peripherals, memory, etc. This assumption was wrong. H. Moore (1965) formulated rule (later called Moore's Law), under which the performance of computer systems is doubling every eighteen months. H. Moore led his empirical law by constructing the dependence of the number of transistors in integrated circuit from time to time. As a consequence of this law we can derive the rate of miniaturization of the individual transistor. The development of digital electronic technology leads to the fact that the size of an elementary computing device is approaching the size of molecules or even atoms.

At this level begin to operate the quantum laws, which for many important dynamic problems have not been described theoretically. Increase the speed of computing devices and reduce their size inevitably leads to the necessity of operations with "transitional" processes. Instead of primitive operations with classical bits, it would be natural in the future to turn to the operations specified by those or other dynamic models of the microworld. Under the ''primitive" model of computation, one can, in particular, understanding and execution of operations with classical bits.

Of course, the introduction of a broader class of primitive models would be more justified if it were possible for a function whose values are recorded in the quantum register to define the operation, and implement effectively, for example, Fourier transform. And it may be a real suggestion that the time for its implementation will be quite comparable to the time of execution of one of the classical operations, as well as operations such as ''convolution" functions could easily be found in the ''nature". Recent studies of similar models show that their performance at the expense of the inherent nature of the capacity for self-organization is not necessarily ''expanded" into simpler components, i.e. not always be written in the form of the classical algorithm.

Series of results obtained in the theory of complexity of algorithms supports the theoretical inability to effectively address many important from a practical point of view tasks. In particular, tasks related to the modeling of natural processes. This means that when you try to descriptions of the process under a given accuracy with the help of a Turing machine of its alphabets of states and memory and/or time of operation are unacceptably high. One of the first of these theoretical difficulties facing developers of automatic control systems, which conduct research at the intersection of informatics and a specific scientific discipline (physics, biology, chemistry, etc.). In scientific literature some authors express doubt in correctness of Church-Turing thesis (see, e.g., Doyle, 1982; da Costa & Doria, 1991; Hogarth, 1994; Copeland, 2004). As example, Penrose (1989) claims that the present computers unable to have intelligence because it is an algorithmically deterministic system. Because things such as the insolubility of the halting problem and Gödel's incompleteness theorem restrictions, he notes that a process can conceivably be deterministic without being algorithmic. Following Penrose, no one can create an artificial intellect, using deterministic computing devices, even if the emergence of quantum computers would lead

to a fundamental breakthrough in the field of computer technology. The only way to compute a complex system without reduction — is to emulate such system on another machine using computational primitives (ideal case imply device, emulating faster than original system processing).

At the same time the whole range of modern methods of control theory does not fit into the discrete model of the classical TM. One possible way of generalization — is the use of so-called "hybrid systems", whose properties are actively studied in recent years (see, e.g., Gao, Lygeros, & Quincampoix, 2007; Goebel, Sanfelice, & Teel, 2009). In the practice of automatic control has been a fruitful idea of relay control, consisting of switching control system with one "sliding" mode to another. In the last decade in systems management is widely implemented neural network approach. It is often proposed to use in the control circuit of neurocomputers. But the works of these devices are not described fully classical scheme of the Turing machine.

One option for expanding Church-Turing thesis is the introduction of a new concept of computing devices, in which the calculation are not built on the traditional algorithms, and on models. The next natural step — a synthesis of models of dynamic systems. We should replace the notion of primitive as a simple mechanical action (e. g., movement of abacus knuckles) to the complex nonlinear physical process, in particular, to evolution of certain quantum mechanical system. Accordingly, we need to reconsider the notion of the result of such action. Instead of an explicit provision "knuckles", one should consider uncertain state of a quantum mechanical system. Evolution enables an exponential speed up of information processing. Quantum measurements bring in probability (stochastic) nature of the understanding of the results of the computational module. To emulate such the measurements we need to consider stochastic jumps.

The main (and most controversial of the modern point of view) notion of a discrete scheme of the Turing machine is the concept of "step". His deeply contradictory messages were seen by the ancient Greeks. One of the possible equivalents of his language is a famous paradox of Zeno's Achilles and the tortoise. In classical set theory, he serves as the insolubility of the continuum in the Zermelo-Fraenkel axiomatics. According to contemporary commentators Zeno really trying to prove that "logic and common sense are incompatible" and that strict logical reasoning does not apply in real life.

Another serious limitation of the classical model of computation is the division of memory cells to isolated bits, because, firstly, a reduction in the length of stroke and distance between bits with a certain level makes it impossible to consider them in isolation because of the laws of quantum mechanics, and secondly, the rejection of "scalar bits" suggests in principle the possibility of the multidimensional (vector) operations in a "step". For example P. Shor (1997) proposed algorithm for a quantum Fourier transform, which can be performed during the time, proportional to $(\log N)^2$, and not $N \log N$ as the classical fast Fourier transform. Tien (2003) discusses based on quantum adiabatic theorem hypothetically possible ''physical'' way to solve at finite time the 10-th Hilbert problem. In Vakhitov, Granichin, and Sysoev (2006) it was sugested the new effective SPSA-type algorithm which computes the estimate of the multidimensional function gradient vector on one "step", or "cycle".

In Granichin and Zhuvikina (2006) it was proposed and justified the new abstraction scheme of computing devices, which includes all of the above approaches. The new concept is fundamentally different from the classical inclusion of "continuity"' of evolution and rethinking the concepts of "tape" and "memory cell". If in the classical approach, memory cells are used to store digital data and change is possible only when the tape shows a pointer to it, the new concept of "memory cell" is a permanently functioning model of a dynamic system (possibly quite complicated), and "tape" is a memory, which has a heterogeneous nature, which may be used are common to all cells of

the program evolution. Obviously that classical memory cell to store the "bits" is a special case of this generalization.

In any modern computer information storage devices based on those or other physical principles, only the evolution of the cell during storage more simple — remains a constant sort of physical characteristic. In other words, the traditional computer science can be compared with the arithmetic, it operates with numbers — the values of memory cells (or, more accurately — the arithmetic of finite binary fractions). The proposed new model of computing will go to the computer from the "arithmetic" to "functional analysis", which operates under the processes of evolution of information within the new "cells".

In the 60 years of the last century against the backdrop of rapid progress in the development of electronics technologies for very large scale integrated circuits (VLSI), cybernetics promised from day to day to give the task of creating "artificial intelligence". But soon supervening frustration was due to unresolved issues about the practical possibility of its creation. Moreover, many modern studies have shown the futility of work in this direction by using the classical Turing machines. In a sense we can speak about the discrediting of the term "artificial intelligence". Likely significant progress in understanding the precise formulation and solution to the problem of creating an artificial intelligence can not be achieved without a rethinking of the traditional answer to the question: what is the calculation process and what is the result of calculations?

Paper organized the following way. In the next section the computability of some problems is discussed. The third section is devoted to the description mathematical model of the new computational processes as a set of parallel working dynamic systems with common parts of state space (shared memory) and "explosive" switches between them. In the fourth section provides a formalized description of a generalized Turing machine, which includes the refinement of earlier attempts to describe such a system (Granichin & Zhuvikina, 2006). Further demonstrates the versatility of the new model through the description of classical Turing machine on generalized.

Computability and Computable Object

To date, there is a sufficient number of examples to prove illusory belief that the solution to a complicated dynamic problem with the desired accuracy can be obtained using the classical Turing machine (Turing, 1936). In astronomy, the laws governing the movement of celestial bodies are well known - this is Newton's law of gravitation or Einstein's equations specifying it. It would seem that the more accurately we define the initial conditions — position and velocity of celestial bodies at the initial time and make the smaller time step for integrating the equations of motion of the engine, the more accurately we can predict the behavior of the system in the future. However, in this way still can not answer the fundamental question of the stability of the solar system at large times. This means that even a small unavoidable inaccuracy of the initial data makes the system behavior unpredictable: the apparent simplicity of the equations of motion, however, conceal their non-integrability, any small inaccuracy in the initial conditions greatly affects the subsequent evolution.

One of the examples from biology shows that even the exact assignment of initial conditions does not guarantee that the dynamic problem. As is known, the exponential nature of change in population size of any species with a constant coefficient of growth is valid only in the absence of limits to growth of the population (e.g, with an unlimited supply of food and the absence of external enemies). If the limits of population x_n growth after n years with the initial value x_0 satisfies the following equation (Verhulst process)

$$x_{n+1} = f(x_n) = (1 + r)x_n - rx_n^2,$$

where r is a constant related to the population growth rate for the year. It turns out that the behavior of the system obeying such an equation, predictably, only for small values of r. When $r = 2.3$, population size begins to oscillate periodically between values 0 and 1. With further increase of r in the value behavior of the population size is complicated. First, begin fluctuations in the number between the four characteristic values with a doubled period, then - the number of values between which the oscillation of populations, doubling with a doubling of the period of oscillation. Finally, when $r = 2.57$, the process ceases to be periodic — place the number of hops between an infinitely large number of values. Thus, the prediction of system behavior is impossible, despite its absolute determinism and certainty of the initial conditions. Behaviors are chaotic, i.e., There is no way to predict the development of a system for a long time. Reason is the nonlinearity of the equation of Verhulst.

The idea of computability implies a requirement to find the approximation for a given object with any desired accuracy in the class of computable objects. Computable objects in the sense of classical Turing machine is a set of finite binary fractions. Thus there is a reduction of the complexity of the object being studied — for example, the irrational or transcendental numbers are replaced by much more simple computable objects — finite binary fractions.

In recent decades, this approach has insurmountable difficulties in describing the specific natural systems (namely, the so-called "complex systems"). These include, inter alia, physical environment, with a complex, in some sense pathological, morphology — spongy, branching, scaly, colloidal media, etc. As a specific policy challenges that arise when describing them, we mention the divergence of the thermodynamic pressure in very small pores, the inability to find a highly developed area of the surface of the gel: the smaller the scale of measurement is taken, the more we obtain the desired value. Similar difficulties arise when measuring the length of the coastline or determining the amount of clouds. Such systems are called fractals (Mandelbrot, 1982), they are intermediate objects between point and line, line and surface, surface and volume.

Another difficulty appears (discussed in Granichin & Khantuleva, 2004) when one try to make formal description of dynamic "transitional" processes in the systems arising from the rapid change in the "external" conditions. Conventional approaches to the description of nonequilibrium processes are inefficient, because the structure of the state space depends on time. Many experimental observation of the occurrence of nonequilibrium processes in the system confirms the emergence of new structures of mesoscopic (intermediate between the micro and macro) scale, which to a large extent determine the dynamic change in the type of formal model. Examples of this kind of structure can be: the clustering in the flow of concentrated disperse mixtures, the formation of multi-scale vortex structures in turbulent flows of liquid and plastic flow of solids under impulsive loading, as well as the hierarchy of structures in living systems. It is now known that the synergistic processes of forming dynamic structures in mesoscopic open thermodynamic systems associated with the emergence of information and management feedback, internal governance, which, together with an external control imposed on the system through the boundary conditions and leads to a discretization of space and time non-equilibrium systems. In this case the physical carriers of information are the elements of dynamic structures. Another example of such a systems can be the Navier–Stokes equations from continuum mechanics, describing the motion of fluid substances and other macroworld models with "jumps" and complex states.

To describe the physical processes in the evolution of complex material systems, the traditional approach to the reduction of complexity finite binary fractions is equivalent to using two models of the material environment — a material point and Jordan domain in Euclidean space. These models are a reflection of the mathematical definition of additive physical quantities (mass, energy, electric charge) only as purely

atomistic or continuously distributed measures. Undecidability of the continuum in the classical set theory is, ultimately, the reason for not allowing to give a closed description of the properties of real physical systems associated with the change of form and energy dissipation in the traditional approach to the concepts of computability and computable objects used in the traditional Turing machine.

The proposed new approach is considered further that the complexity of a computable object should be equivalent to the complexity of a real object under study and adequately reflect its properties. For the above material media it will be the ability to flow, change shape, to shrink/stretch, to break up/merge. To do this you must use a mathematical model of the material environment, based on the weaker, compared with the requirement to be a system of isolated points or Jordan area requirements. Namely, the simulator set must be closed (i.e., contain all its limit points) and dense in itself (i.e., each point should be the limit). Joint performance of these properties is equivalent to the property to be a perfect set. There are many types of perfect sets, among which there are regular, verifiable formal construction and description. Some of perfect sets have properties in a very specific sense, close to the properties of sets of isolated points, and some — to the properties of Jordan domains. In intermediate cases, they reflect well on the properties of fractals and are measurable in the sense of p-Hausdorff measure (Hausdorff, 1919). Viability of the concept of Hausdorff dimension due to the fact that this dimension, regarded as the present continuous variable, may be a characteristic that describes the evolution of a complex system including the energy dissipation, while the topological dimension of Urysohn-Menger in the process of change real system, typically remains constant (Zhuvikina, 1997, 1998). Moreover, requires the synthesis and the notion of computability. Let's illustrate a number of examples.

Example 1. The square root of 2. By definition, $\sqrt{2}$ is the length of the diagonal of the unit square.

We believe that this number is calculated, if built unit square. We say that the complexity of $\sqrt{2}$ is equal to the complexity of the unit square. Unit square called computational primitives for $\sqrt{2}$ and arithmetically related irrational numbers.

Example 2. Measuring snowflakes. Snowflake is inscribed in a circle of radius 1 cm with a given fractal morphology and Hausdorff dimension 5/2 with six rays.

As the size of the snowflakes accepted value 6 $cm^{5/2}$. Fractal dimension and morphology of this is a computational primitive for the measurement of the snowflake.

Example 3. For Diophantine equations with rational integral numerical coefficients indicate the method by which possibly after a finite number of operations to determine whether this equation is solvable in rational integers.

Unsolvability of the problem on the classical TM was shown in Matiyasevich and Robinson (1975). Tien (2003) discusses hypothetically possible "physical" way of solutions, based on the quantum adiabatic theorem.

Thus, the object believe (cognitively) computable if it is arithmetically related to the presence in the memory of the computing device corresponding computational primitives. Computational primitive is one of the basic concepts of the proposed new approach to the construction of a generalized model of the Turing machine.

THE INTERACTION OF PARALLEL DYNAMICAL SYSTEMS WITH SHARED MEMORY

Let's define a computational process as a collection of working in parallel dynamical systems with overlapping parts of the state space (shared memory).

Consider a computer system consisting of a countable set of analog component $\{M_i\}_{i=1}^{\infty}$, such that each component M_i determines the dynamics of some components x_i of shared memory x (which has some heterogeneous nature) by the rule

$$\dot{x}_i(t) = H_i(x_i, t),$$

where $x_i(t) \in \mathrm{X}_i, \mathrm{X}_i \subset \mathrm{X}$. Suppose that at any given time t only a small part of the models (a computing device component) has an impact on the change of memory state on all system. In other words, at each moment t one can define an infinite string of bits $S(t) = \{s_i(t)\}_{i=1}^{\infty}$, consist of a finite number of units: $s_i(t) \in \{0,1\}$, for $i \le i_{\max}(t)$ and $s_i(t) = 0$ for $i > i_{\max}(t)$. Then the general dynamics of the system can be described as

$$\dot{x}(t) = \sum_{i=1}^{\infty} s_i(t) H_i(x_i, t),$$

where $x(t) \in \mathrm{X}$ and $x_i(t) \in \mathrm{X}_i$.

For macroworld models, apparently, x would have real values ($x(t) \in \Re$), but each model operating on the molecular and atom level may pose as a Schrödinger equation, and operate under complex space, so $x(t) \in C$.

Line $S(t)$ in some sense can be considered analogous of the state of the classical Turing machine (TM). Denote the set of all possible states as Σ.

Without loss of generality, we may assume that $S(t)$ — a piecewise-constant function of time, i.e. at certain intervals or that the model (the components of computational device) "involved". Thus obtain a hybrid dynamical system in the traditional sense of the term. The overall dynamics of the system on k-th constancy interval of $S(t)$ from t_k up to t_{k+1} (with the value of S_k) can be described as:

$$\dot{x}_{S_k}(t) = \sum_{i=1}^{\dim(\bar{\mathrm{X}}_{S_k})} s_i(t_k) H_i(x_i, t) = \sum_{i \in \mathrm{sup}\, p(S_k)} H_i(x_i, t), t \in [t_k, t_{k+1}),$$

where $x_{S_k}(t) \in \bar{\mathrm{X}}_{S_k} = \bigcup_{i \in \mathrm{sup}\, p(S_k)} \mathrm{X}_i \subseteq \mathrm{X}$.

For a complete description of the entire system dynamics we must determine the rule of changing $S(t)$ and the rule of termination. Describing the processes of solving (or simulation) large number of practical problems is sufficient to use the directed graph of transitions linking the possible transitions $S_k \xrightarrow{J_{k,j}} S_j; S_k, S_j \subset \Sigma$ with the conditions $J_{k,j} \in J$, under which the switches change the dynamics of the entire system with a set of models S_k to S_j.

Thus we can determine that in a new sense, the program P is the rule that change $S(t)$. If the rules are stored in the memory of a computational device and enable their adjustment over time, from the standpoint of the programming is in some sense be consistent with the von Neumann architecture, where both data and program stored in a memory and program changes during system operation.

Condition of system termination is $S(t) = 0$, which is equal to the stop of all the memory change processes a way that not any of the model components are active.

Example 4. Consider the classical description of the TM in the presented model.

Turning to the classic machine, every cell of the tape can be regarded as a model that takes values on a finite alphabet. Entire tape of this machine can be considered as a computing system consisting of working in parallel dynamical systems. In this case, machine program can be represented as directed graph P,

nodes of which are $S_{j,k}$ and edges are change conditions $J_{j,k}$.

If we index the cells using $k \in N$, then the machine at each operation step can be described by a simple discrete system

$$\begin{cases} x_{k\pm1}^{t+1} = f_{k\pm1}^{t}(x_k) \\ x_k^{t+1} = f_k^{t}(x_k) \end{cases},$$

where x — is a vector that stores states of the machine, and the content of the memory.

At each step of the classical Turing machine will be exactly one active cell of the tape, that is, each S_k will contain only one unit. Accordingly, only one of the three functions $f_{k,k\pm1} \neq 0$.

The work of such a system is illustrated in Figure 1.

Here S_I and S_F — initial and final states. $S_{k-1} = (0,0,0,..1,0,0,...0,0,0)$, $S_k = (0,0,0,..0,1,0,...0,0,0)$, $S_{k+1} = (0,0,0,..0,0,1,...0,0,0)$, such that ones corresponding to the k, $k-1$, $k+1$ cells of the tape.

Let's mention that in the real realization of the machine, there must be time-cycling device $x_t(t) \in \{0,1\}$.

Example 5. Consider a system consisting of three models $x_{1,2,3}$ (under the model implies a complex system — memory and state), which makes quality jumps from initial state $S_I = (1,1,1)$ to $S_1 = (0,1,1)$, or $S_2 = (1,0,0)$ states, and from them to the final state $S_F = (0,0,0)$ (if it matches the conditions $J_{I,1}, J_{I,2}, J_{I,F}, J_{1,F}, J_{2,1} \in J$). Let also $x_{1,2,3}$ share a common, 2-component memory $X = \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix}$ in such a way that $x_1 = (\tilde{x}_1)$, $x_2 = \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix}$ and $x_3 = (\tilde{x}_2)$.

Such a system can be represented by graph (Figure 2).

In the S_I state model can be described by the system:

$$\begin{cases} \dot{x}_1 = f_{I,1}(\tilde{x}_1, t) \\ \dot{x}_2 = f_{I,2}(x_2 = \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix}, t) \\ \dot{x}_3 = f_{I,3}(\tilde{x}_2, t) \end{cases}.$$

Similarly, for the state S_1:

$$\begin{cases} \dot{x}_1 = 0 \\ \dot{x}_2 = f_{1,2}(x_2 = \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix}, t) \\ \dot{x}_3 = f_{1,3}(\tilde{x}_2, t) \end{cases} \quad \text{and } S_2: \begin{cases} \dot{x}_1 = f_{2,1}(\tilde{x}_1, t) \\ \dot{x}_2 = 0 \\ \dot{x}_3 = 0 \end{cases}.$$

In the final state S_F dynamics of the system is absent, i. e., $\dot{x} = 0$.

Step of the system is measured at the time of entering the terminal set J. The above equations describe the interstep dynamics.

More general scheme is obtained when setting the rules of stochastic transitions. Described deterministic case are well generalized by setting a sufficiently high probability of conversion for some of its transitions and leaving equiprobable, almost zero probability, level for all the others. But even this scheme can be more

Figure 1.

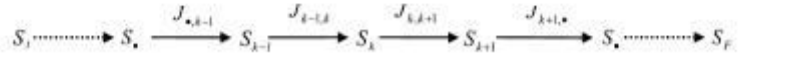
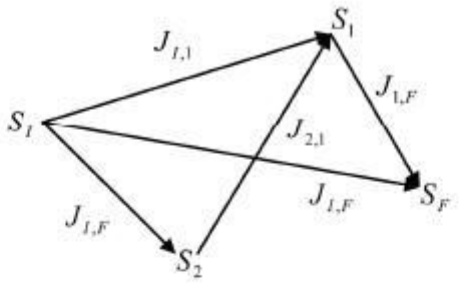


Figure 2.



effective (by analogy with the known method of annealing to find the global minimum of the function). Accidentally set "new relationship" may be useful for the system and can prevent the adjustment of the transition probabilities in certain situations to enhance their value. Such a mechanism would lead to elements of self-organizing models in the process.

Note the fundamental difference between the concepts of state and memory, absent in most of the actual computing devices. Memory assumed divided into individual cells, the content of whose varies, generally speaking, independently of each other. State on the other hand — is an integral characteristic of the machine, possibly, affect the contents of memory. The result of computation in the Turing machine at the end of work is exactly the content of memory. As well, dividing memory and state has the practical reasons — as memory and state appears as a different "entities" in real world devices, it's easier to work with them apart.

GENERALIZED MODEL OF THE TURING MACHINE

To overcome the above limitations can model a generalized Turing machine (hereinafter, GTM). The machine requires a heterogeneous memory and the set of states, which vary during operation. In addition, operators are given the evolution of states and memory.

So, we define the GTM as a structure

$$\langle S, s, s_0, \mathrm{X}, x, x_0, E, J, P, F \rangle$$

in which: S is a set of states; s is the current state of the machine; s_0 is the initial state; X is a set of memory states; x is the current memory content, where $x \in \mathrm{X}$;

x_0 is an initial contents of memory; $E : Q \to Q$ is an operator of the evolution of states and memory, where $Q = (S, \mathrm{X})$ — the set of all possible combinations (s, x) of states and memory;

$P : J \to Q$ is a machine program, where $J \subset Q$ — domain set, and $J \cap F = \varnothing$; $F \subset Q$ is a final set of states and memory.

In the classical Turing machine memory (tape) — is an ordered set of cells, each of which contains values of some alphabet Γ. Tape concept in GTM converted to the notion of memory x, which has a heterogeneous nature. Alphabet for the symbols, which are located in each cell of the TM, is an arbitrary memory state space X.

Memory x is evolving under the action of the operators E and P from the initial state x_0 until the current state and the memory of the machine (s, x) does not appear in the final set F. In this case, the machine stops and the current state and the memory is the result of work. If the memory and the state falls into

the terminal set J, then machine jump by the program P, and then continues to evolve under the operator E until it re-entering J.

We described memory and states as different entities because they are two different abstractions. For example, in some computing devices memory assumed divided into individual cells, and state is an integral characteristic. Other devices have memory integrated with states. Using GTM one can describe all the types of such computational devices. At the same time defined model allow us to consider memory and state as a complex state with tuple $\langle Q, q, q_0 \rangle$, where $Q = \langle S, X \rangle$, $q = \langle s, x \rangle$, $q_0 = \langle s_0, x_0 \rangle$.

Another reason is that the model must safe succession with classical TM where memory is a tape and state is integral characteristic of automata.

Cycle system work, in contrast to the classical TM, where the cycles are measured shifts of the head, is considered a variable interval of time between hitting the memory and the state machine in the terminal set J.

To describe parallel processes beneficial to consider the set of machines, where a single machine (hereinafter, "model" or "evolutionary primitive") appears as

$$M^j = \langle S^j, s^j, s_0^j, \mathrm{X}^j, x^j, x_0^j, E^j \rangle,$$

where separate for each machine state $s^j \in S^j$ and memory $x^j \in \mathrm{X}^j$, evolving under the action of the operator E^j.

In this case, shared memory machines is a countable set $\left\{x^j\right\}_{j \in \mathbb{Z}}$. Thus for all models is given general program P, that works for the entire memory at once and takes into account the state S_j of each model.

Note also that the early defined model (Granichin & Zhuvikina, 2006) has the structure

$$\langle S, s, s_0, Y, X, x, x_0, F, J, P, E \rangle,$$

where Y is a tape (memory), X — set of memory states, x — the current memory content: $x(y) \subset X, \forall y \in Y$; x_0 — the initial contents of memory; the terminal set $F \subset Q$, where Q — the set of all pairs $q = (s, x)$ of states and memory; $J \subset Q$ — domain for program, and $F \cap J = \varnothing$; $P : J \to Q$ — program; $E : Q \to Q$ — operator of states and memory evolution.

As you can see, in this article authors could exclude a pointer x to the memory components and describe memory as an object of fully heterogeneous nature (or, in the case of memory divided into cells — in the form of a Cartesian product component). In this case for the proposed model remain true all allegations of the old model (since you can always use the coordinate functions for the corresponding components) and, inter alia, are faithful to all previously described examples of effectively computable functions.

Note that we could determine machine shutdown moment only by the value of the state, but the introduction of a more general rule, obviously, includes this case. In the new model excluded the notion of a working cell, i.e. change of state may depend on the contents of all memory at the moment, but the contents of memory can be changed simultaneously in the whole space X.

Separation of the changes description in the state and memory into two components: the evolution E and program P, at first glance, may seem artificial. In fact, this separation allows the lines of demarcation "forced" changes are usually made to the system from outside, asked by someone "knowingly" and those "natural" processes that occur in some physical (or biological) system by the laws of nature.

Another generalization of the Turing machine can be probabilistic mappings of operators P, and E that will sell through a new model of dynamical systems can not be described by deterministic laws, as well as stochastic hybrid systems, probabilistic automata, systems with stochastic control, etc.

Let's define program P as a stochastic function $\tilde{P}_{kl} : J_k \xrightarrow{\alpha_{kl}} J_l$, where $J_k, J_l \in J$, and α_{kl} is a probability of jumping from state-memory combination J_k to the state-memory combination J_l such that $\sum_{k,l=0}^{\infty} \alpha_{kl} = 1$ (infinite case). In quantum mechanics this "stochastic jumps" can be compared to the measurements. Between a measurements (steps), quantum system evolving by the action of evolution operator E.

In many cases, randomization in many complex cases isn't complicate the rules of the work, but allowing to compensate the inadequacies of the mathematical description of the dynamics of processes and events. Randomization allows to partially eliminate the influence on the work of systematic errors (Fedin, Granichin, Dedkov, Yu, & Molodtsov, 2004, 2008), which are almost inevitable in changing over time models of dynamic systems. Moreover, one can find in (Granichin & Khantuleva, 2004) an example of effective use of randomized policy impacts on the system in a dynamically changing structure of the state space.

DESCRIPTION OF THE CLASSICAL TM USING GTM

Hopcroft and Ullman (1979, p. 148) formally define a (one-tape) Turing machine as a 7-tuple

$$M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle,$$

where

Q is a finite set of states;
Γ is a finite set of the tape alphabet/symbols;
$b \in \Gamma$ is the blank symbol (the only symbol allowed to occur on the tape infinitely often at any step during the computation);
$\Sigma \subseteq \Gamma \setminus b$ is the set of input symbols;
$\partial : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is a partial function called the transition function, where L is left shift, R is right shift;
$q_0 \in Q$ is the initial state;
$F \in Q$ is the set of final or accepting states.

Anything that operates according to these specifications is a Turing machine.

The algorithm works consist in the fact that at each cycle in accordance with the rule changes the contents of working location and condition, as well as the shift of the tape. If the new state q of the machine falls into the final set F, then the machine stops and the result of her work is considered to be the current content of memory. In addition, the equivalent would be description of the machine in which instead of tape shifting, the pointer shifting $x = x \pm 1$ to the current working cell happens.

Let's show how one can describe such a system using the generalized TM. Tape corresponds memory $x = \Gamma^{\infty}$, where Γ — a finite alphabet of classical TM states. Cells are numbered by indices $j \in \mathrm{Z}$, and x_j — value of the cell j, such that $x_j = \phi_j(x)$, where ϕ_j — the corresponding coordinate function. Current state $\tilde{q} = \langle q, k, t \rangle$ of generalized machine stores the index $k \in \mathrm{Z}$ of the current working location (position of read/write heads), the proportion of time $t \in [0,1]$ within the cycle length δ, and q — the status of classic machines. Accordingly, the state space $\tilde{Q}$ is described as $Q \times Z \times [0,1]$ and the initial state as $\tilde{q}_0 = \langle q_0, 0, 0 \rangle$, i.e. in the initial state head pointing to the zero working cell.

In addition, the final set $\tilde{F} = \langle F, X \rangle$ and the set J — domain of program P is defined as a subset $\langle \tilde{Q} \setminus \tilde{F}, \mathrm{X} \rangle$.

At the transition from i to $i+1$ cycle, tape left/right shift is simulated by the pointer to a working cell shifting: $k_{i+1} = k_i \pm 1$. This GTM program P is a piecewise-defined function, where its components

$$P_j(x_i,\langle k,q\rangle) = (\tilde{x}_i,\langle k+\begin{cases}\pm 1\\ 0\end{cases},q\rangle),$$

where $P_j : J \to \tilde{Q}$. Evolution operator E uniformly alters the time $t : \dot{t} = 1/\delta$, and is identical to the other components of the status and contents of memory.

DESCRIPTION OF THE MOORE MACHINE USING GTM

Let's show how each of the Moore automata (or it corresponding Mealy machine) can be defined using GTM.

Moore finite state machine is a finite state transducer (automata with two tapes — input and output) that can be defined as a 7-tuple

$$\langle S, s, s_0, \Sigma, \Lambda, T, G\rangle,$$

where:

S is a finite set of states;
$s \in S$ is a current state;
$s_0 \in S$ is an initiate state;
Σ is a finite set called the input alphabet;
Λ is a finite set called the output alphabet;
$T : S \times \Sigma \to S$ is a transition function which mapping a state and the input alphabet to the next state;
$G : S \to \Lambda$ is an output function mapping each state to the output alphabet.

Corresponding GTM would be 10-tuple $\langle S, s, s_0, X, x, x_0, E, J, P, F\rangle$. States of this machine is equivalent to Moore machine $\langle S, s, s_0\rangle$ — set of states, current state and initiate state. Memory $X = \langle \Sigma, \Lambda \cup \{e\}\rangle$ is two-component. Here Σ and Λ are GMT equivalents of the transducer input and output tapes, and e is a special "empty" element. Initial memory state is $x_0 = \langle \sigma \in \Sigma, e\rangle$. GTM evolution operator E is constant, and program P is a piecewise-defined function

$$P = J \to J \cup F,$$

where P represent both, transitional and output functions of Moore machine, and output function push automata to the final state $F = S \times \langle \Sigma, \Lambda\rangle$. In addition, jump set J include each memory and allowed machine states except final, so $J = S \times \langle \Sigma, e\rangle$.

EVOLUTION OF THE MODEL

The proposed model of computing allows one to describe, if not all, the vast majority of the processes of the real world, as well as the work of various existing and future computing devices, including analog and biocomputers, neurocomputers, quantum computers, etc. The peculiarity of the proposed approach is the rejection of the reduction of complexity in the process of calculation. The complexity of a computable object should be the equivalent of a calculated.

The complexity of the proposed model is calculated on the basic primitives. In the classical algorithms calculation of the function occurs through sequential or parallel application of the simplest action in each cycle (or step). In the proposed model it's possible to calculate within one step more complex functions (such as convolution or Fourier transform, or even dynamical system), if there is matching evolution primitives. This allows to look at more comprehensive theory of computability based on evolution primitives, rather than on traditionally considered bit transformations {0, 1}.

Moreover, proposed GTM model designed from the hopping nature of quantum computation. Considering quantum algorithm consisting of the sequence of the transforms, and measurements — for example, Grover's (1996) algorithm for searching an unsorted database consist of uniforming distribution over all eigenstates, performing sequence of Grover iterations (using two unary operators), and making measurements. Here first and second steps of the algorithm can be made using evolution

operator, and measurements — using stochastic program operator.

Proposed model about the same time should not lose the generality, that is, it must describe computations similar to the produced by the classical Turing machine. Quantum and neurocomputers promise to drastically change the picture of the computational power of modern computing devices. Increased computing power, possible through the use of new computing models based on physical phenomena, suggests that in the future, new computers can solve problems impossible for ordinary computers.

REFERENCES

Church, A. (1936). A note on the Entscheidung problem. *Journal of Symbolic Logic, 1*, 56–68.

Copeland, J. B. (2004). The Church-Turing thesis. *NeuroQuantology, 2*, 101-115.

da Costa, N. C. A., & Doria, F. A. (1991). Classical Physics and Penrose's Thesis. *Foundations of Physics Letters, 4*, 363-374.

Deutsch, D. (1985). Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. In *Proceedings of the Royal Society: series A*, 97-117.

Doyle, J. (1982). *What is Church's Thesis? An Outline. Laboratory for Computer Science.* Cambridge, MA: MIT.

Fedin, D. S., Granichin, O. N., Dedkov, Yu. S., & Molodtsov, S. L. (2008). Method of measurements with random perturbation: application in photoemission experiments. *The Review of Scientific Instruments*, 79.

Feynman, R. (1982). Modeling of physics on computers. *International Journal of Theoretical Physics*, 21.

Gao, Y. G., Lygeros, J., & Quincampoix, M. (2007). On the reachability problem for uncertain hybrid systems. *IEEE Transactions on Automatic Control, 52*(9), 1572–1586. doi:10.1109/TAC.2007.904449

Goebel, R., Sanfelice, R. G., & Teel, A. R. (2009). Hybrid dynamical systems. *IEEE Control Systems Magazine, 29*(2), 28–93. doi:10.1109/MCS.2008.931718

Granichin, O. N. (2004). Linear regression and filtering under nonstandard assumptions (Arbitrary noise). *IEEE Transactions on Automatic Control, 49*(10), 1830–1835. doi:10.1109/TAC.2004.835585

Granichin, O. N., & Khantuleva, T. A. (2004). Hybrid systems and randomized measuring in nonequilibrium processes. *Differential Equation and Control Processes, 3*.

Granichin, O. N., & Zhuvikina, I. A. (2006). New computational model based on evolution primitives: generalization of the Turing machine concepts. *Neurocomputers: design, application, 7*, 24-31.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceeding of the 28th Annual ACM Symposium on the Theory of Computing* (pp. 212).

Hausdorff, F. (1919). Dimension und ausseres. *Math. Ann, 79*.

Hogarth, M. L. (1994). Non-Turing computers and Non-Turing computability. *PSA, 1*, 126–138.

Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to Automata Theory*. Languages, and Computation.

Mandelbrot, B. B. (1982). *The Fractal Geometry of Nature*.

Matiyasevich, Y., & Robinson, J. (1975). Reduction of an arbitrary Diophantine equation to one in 13 unknowns. *Acta Arithmetica, 28*, 521–549.

Moore, H. (1965). *Electronics, 38*.

Penrose, R. (1989). *The Emperor's New Mind: Concerning Computers, Minds, and The Laws of Physics*. New York: Oxford University Press.

Shor, P. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing, 26*, 1484–1509. doi:10.1137/S0097539795293172

Tien, D. K. (2003). Computing the non-computable. *Contemporary Physics, 44*, 51–71. doi:10.1080/00107510302712

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, 2*, 230–265.