The Concept of a Universal Learning System as a Basis for Creating a General Mathematical Theory of Learning

YURY P. SHIMANSKY
Harrington Department of Bioengineering, Center for Neural Interface Design, Arizona Biodesign Institute, Arizona State University, Tempe, AZ 85287, USA;
E-mail: Yury.Shimansky@asu.edu

Abstract. The number of studies related to natural and artificial mechanisms of learning rapidly increases. However, there is no general theory of learning that could provide a unifying basis for exploring different directions in this growing field. For a long time the development of such a theory has been hindered by nativists' belief that the development of a biological organism during ontogeny should be viewed as parameterization of an innate, encoded in the genome structure by an innate algorithm, and nothing essentially new is created during this process. Noam Chomsky has claimed, therefore, that the creation of a non-trivial general mathematical theory of learning is not feasible, since any algorithm cannot produce a more complex algorithm. This study refutes the above argumentation by developing a counter-example based on the mathematical theory of algorithms and computable functions. It introduces a novel concept of a Universal Learning System (ULS) capable of learning to control in an optimal way any given constructive system from a certain class. The necessary conditions for the existence of a ULS and its main functional properties are investigated. The impossibility of building an algorithmic ULS for a sufficiently complex class of controlled objects is shown, and a proof of the existence of a non-algorithmic ULS based on the axioms of classical mathematics is presented. It is argued that a non-algorithmic ULS is a legitimate object of not only mathematics, but also the world of nature. These results indicate that an algorithmic description of the organization and adaptive development of biological systems in general is not sufficient. At the same time, it is possible to create a rigorous non-algorithmic general theory of learning as a theory of ULS. The utilization of this framework for integrating learning-related studies is discussed.

Key words: computability, creativity, physical constructiveness, recursive function theory, theory of learning

1. Introduction

There are at least three main reasons for studying mechanisms and processes of learning. First, we need to understand the nature of learning processes in order to develop better educational programs, help people with learning deficits, and develop further our learning capacities. Second, the creation of

Minds and Machines **14:** 453–484, 2004.

intelligent machines capable of adapting to a complex, changing environment and knowledge acquisition requires a rigorous and powerful theory of learning systems. Third, from a basic-research perspective, the capacity to learn and adapt is one of the most important, essential features of all biological organisms, particularly those endowed with a developed nervous system. In biology, this feature is no less fundamental than, for instance, mechanical properties in physics.

Learning-related phenomena are studied in many fields of scientific research such as psychology, linguistics, neurophysiology, genetics, and artificial intelligence. Each field has its own experimental basis, terminology, and theoretical concepts, which develop mostly in parallel, without much of interaction. At the same time, there obviously is a need for exchanging useful terms and concepts between these fields. For instance, so-called genetic algorithms become increasingly popular in the computer modeling of learning in neuronal nets. The nets are 'mutating', exchanging 'genetic information' as a results of 'cross-over' operation, etc (Mitchell, 1996). The concept of reinforcement borrowed from the psychology of learning behavior is now extensively used in the 'reinforcement learning' methods in neuronal net models (Barto et al., 1983; Sutton and Barto, 1998). The neuronal net approach, in its turn, was found to be quite useful in the field of language learning (Calvin and Bickerton, 2000).

The amount of learning-related experimental data and models dealing with certain clusters of those data has already become almost unmanageably large and it is rapidly growing. This situation is not at all new to the history of science. It is well known that a good general theory usually is capable of packing knowledge into a system based on a mathematical description of a relatively small number of concepts and laws. Such a system makes it possible for the students to learn just the concepts and laws themselves, instead of spending years on studying separate facts related to the manifestation of those laws in a huge number of particular cases. Furthermore, it would enable them of not only recreating the known experimental data, but also predicting the results of new experiments. The process of theory creation happened already many times in different parts of science, which by itself is one of the most important experimental facts related to learning. Yet, there is no mathematically rigorous general theory of learning (GTL), which could serve as a unifying and integrative basis for different domains of learning-related research. An obvious question arises, "What prevents one from creating a GTL?" More than 25 years ago, Noam Chomsky, world-known mathematical linguist alleged that there was no formal GTL that could be analyzed based on a mathematical apparatus, and that the creation of such a theory did not seem to be feasible (Chomsky, 1980). What was his reason for such a strong assertion?

This question is directly related to the old problem of finding the role of innateness and learning in determining the phenotype of an organism (so-called 'nature/nurture' problem). It has been one of the central issues in biology for decades, concerning virtually all parts of biology, from genetics to ethology. The 'nature/nurture' problem is marked by a vast diversity of opinions among scientists, which was fully expressed in 1975 at the Great Debate between Jean Piaget and Noam Chomsky. It was a 4-days meeting during which many most prominent scientists of the world, including Nobel laureates in biology and leading figures in behavioral sciences, philosophy, mathematics, and artificial intelligence, were engaged in vigorous discussions.

From Piaget's point of view, knowledge including new skills and such high-level functions as language can be acquired through learning based on the constructions of sensorimotor intelligence (Piaget, 1980). He viewed the process of learning as an interaction of organic auto-regulating mechanisms with the environment (Piaget, 1972). Although Piaget's reasoning was not supported by a formal mathematical theory, it strongly appealed to the audience's intuition.

Chomsky maintained that language and other cognitive functions were innate (i.e. encoded in human genes). Moreover, he argued against the usefulness of the notion of learning on the premise that the process of developing language and other higher-level functions could be described as the work of an innate algorithm setting parameters in the innate structure of the nervous system. In the case of language, this structure should represent a 'generative grammar', a common semantic root for grammars of different human languages (Chomsky, 1966). According to the nativists' point of view, *nothing essentially new is created during ontogeny*, and any act of creativity (e.g., the development of a new scientific theory) should be attributed to the random process of gene mutations. This, for example, is what Chomsky had to say at the Great Debate in response to Bateson's comment regarding learning in chimpanzees (Piatelli-Palmarini, 1980, pp. 262–263): "... suppose now we went ahead and found out what was involved in his [the chimpanzee's, Y.S.] making the next step, and then we ask, what is the origin of that next thing X that is involved in it? You answer, trivially, that it is genetic determination, and you keep doing it at every point where you get an answer, because there aren't any other possibilities." At that time, no one could find a significant contra-argument.

As one can see, the 'nature/nurture' issue actually is equivalent to the fundamentally important question "*What is the origin of the mind's creativity?*" This problem was not resolved at the Great Debate, and it has not progressed much toward a resolution since then. Despite numerous papers written on it, no agreement has been reached on this issue, although some advancement has been made toward the integration of the two viewpoints. For instance, the concept of an algorithm has been successfully incorporated into the

definition of a teleonomic process (Mayr, 1982), a key notion from the field of genetic epistemology originating from Piaget's studies. At the same time, from Piatelli-Palmarini's statement, in which he expresses his agreement with those who say that the term 'learning' is less than useful in science (Piatelli-Palmarini, 1989), one can conclude that nativists do not feel less strongly about their position. A significant support for this position comes from theoreticians of machine learning who acknowledge that the mathematical theory of learning has been developed thus far as a theory of learning algorithms (Langley, 1996). Nothing essentially new has occurred in attempts to use the recent achievements in modeling the acquisition of a behavior by artificial neuronal networks in order to solve the 'nature/nurture' problem (Elman et al., 1996). Although the authors' arguments are directed against nativism, they demonstrate that a behavior can be acquired based on an *algorithm* of parameter adjustments in a neuronal network, which actually supports the nativists' viewpoint.

It seems impossible to find the origin of creativity based on experimental research alone, without an extensive theoretical investigation. One of the main problems is that the same experimental observation (e.g., on the development of a child's mental or sensorimotor capacities) can be interpreted by developmentalists as an example of 'true' (i.e. creative) learning and by nativists as the work of an innate algorithm setting parameters in the innate structure of the corresponding part of the nervous system. In order to challenge the nativists' arguments on theoretical grounds, the analysis of the 'nature/nurture' issue apparently must be performed at the same level of mathematical rigor by applying the powerful mathematical apparatus of formal logic, which allowed Chomsky to disperse so easily the non-formal, intuitive arguments of his opponents. Since the Great Debate, several papers have been published in which a generalization of certain mathematical aspects of learning has been proposed (Pla-Lopez, 1988; Amari, 1991) although with no visible regard to Chomsky's arguments. After the first version of this paper has been completed, the author became aware of Marcus Hutter's work (Hutter, 2001). Although not explicitly related to the above issue, it provides a strong mathematical background for a GTL (Section 3.2).

It should be emphasized that the main goal of this paper is to demonstrate that a non-trivial, mathematically rigorous GTL can be created and determine its main features, rather than to solve the nature/nurture issue or disprove Chomsky's arguments against learning as a process during which something not innate is created. The problem of creativity origin is regarded here as a beacon that indicates where the initial effort should be directed, for two reasons. First, one obviously cannot avoid the challenge posed by this problem when creating a GTL. Second and more important, one can expect intuitively that systematic efforts to solve such fundamental and difficult problem would provide a substantial insight into the GTL development *per se*.

2. Existing Conceptual Bases

From a very general perspective, learning is a function performed by a physical system, which interacts with a certain environment. Before venturing into mathematical formalization of learning, it seems important to realize that we do not know *a priori* which physical properties of either the learning system or its environment are essential for performing that function. Furthermore, from a physical viewpoint, we also need to explicitly assume existence of an external observer, who performs physical measurements on both above systems and, therefore, unavoidably (from a quantum-mechanical perspective) interferes with their behavior. When abstracting ourselves from these details, we should remember to go back and try reinstating some of them, in case a principal problem is found in the course of theory development.

During a process of learning, the modification of the *function* performed by the learning system is achieved through the modification of the system's *substrate*. The main mathematical approaches to describing these two properties are the theory of optimal control and the theory of algorithms, respectively.

2.1. OPTIMAL CONTROL THEORY

A learning process, being it an adaptation of a population of organisms to a new environments or a child's learning its mother tongue, always is associated with a certain (explicit or implicit) measure of behavior quality that is supposed to be improved as a result of learning. This fact suggests that learning can be described mathematically as optimization. From another perspective, the theory of optimal control is based on the same Hamilton's principle that unifies all parts of contemporary theoretical physics (Lawrie, 1990). It is well known that the more complex the physical system, the more beneficial and important the utilization of this principle. For instance, one does not need to use Hamilton's principle (based on Lagrange's approach) in mechanics when dealing with a mass point - Newton's laws are quite sufficient for calculating the point's trajectory. However, with an increase in the mechanical system's complexity, the differential equation system describing its motion quickly becomes practically unmanageable (e.g., this is true already for a three-segmented limb with seven degrees of freedom in 3-D). At the same time, the formula for the corresponding Lagrangian function usually is incomparably shorter. It should be emphasized that the differential equation system describing the mechanical system's motion can be derived from the Lagrangian *automatically*, just as a function's values can be calculated by substituting concrete values of its arguments in the corresponding formula.

Since biological systems (especially the brain) are the most complex known physical systems, it seems quite logical to try Hamilton's approach (in the form of optimal control theory) when seeking an efficient mathematical model of their behavior. The possibility of modeling the behavior of biological systems based on the concept of optimal control has gained considerable attention from biologists. It has been shown that the neural control of movements can be described successfully as the functioning of an optimal control system (Uno et al., 1989; Johansson and Magnusson, 1991; Shoucri, 1991; Viviani and Flash, 1995; Lan, 1997; Bhushan and Shadmehr, 1999; Shimansky, 2000; Shimansky et al., 2004). The theory of stochastic optimal control (in the form of so-called 'reinforcement learning') has proved also quite successful for modeling the process of learning to play antagonistic games (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998).

2.2. THEORY OF ALGORITHMS

The theory of optimal control does not deal explicitly with the implementation of the functions involved. In a general case, however, the criterion of learning system's optimality can include the *complexity* of its implementation (e.g., the amount of physical matter and energy required). This consideration touches upon a very general principle, the law of parsimony (a.k.a. rule of Ockham's razor), which states that the descriptive complexity of a model of reality should be minimized. This law dictates the development of the simplest theory that fits the entire amount of experimental data and is capable of predicting results of new experiments. This important, although informal idea finds a perfect mathematical formalization in the theory of algorithms and recursive functions as Kolmogorov complexity (Li and Vitanyi, 1997).

The mathematical notion of an algorithm is the result of generalizing the idea of an effective procedure that can be performed in a finite number of steps by executing an elementary operation at each step. It should be emphasized that many different formal mathematical models of an effectively performable procedure have been created, and all of them have appeared to be equivalent to each other (Uspensky and Semenov, 1993). This fact apparently should be considered a result of series of independently performed physical experiments, for as much as the brain is a physical system generating on its output a solution to a given problem. It strongly indicates the fundamental character of the notion of an algorithm. Perhaps the most frequently used mathematical model of an algorithm is a finite sequence of commands (a program) that can be performed by a certain automaton, e.g., a Turing machine, the mathematical prototype of current computers.

For many years after the development of the Turing machine, many scientists have been thinking that the brain itself is a very large natural

computer, ever so complex due to the complexity of the working algorithm. However, over time this identity has become less obvious. Some scientists have argued that while the human brain can learn, a computer cannot. It has been shown that in many cases learning can be modeled successfully by an algorithm (Hebb, 1949; Rosenblatt, 1962; Kirkpatrick et al., 1983; Rumelhart et al., 1986). The most serious arguments against the Turing machine as a functionally complete brain model allude to the capability of the human brain to set up a problem and create new ideas. It also has been suggested that consciousness is the critical function that makes the brain more than a machine (e.g., Penrose, 1994). However, no one has been able to define these functions sufficiently well to compare the functional power of the brain with that of a computer on a rigorous mathematical basis. Thus, the Turing machine, a *universal computer* (i.e. an automaton capable of performing any algorithmic function the description of which is sent to the its input), still is the closest known mathematical approximation of the brain as an information processor.

The universal computer model of the brain's functioning focuses attention on its computational properties. Are these features *sufficient* for understanding the functioning and development of the brain, a super-complex physical system continuously interacting with the environment? Many additional questions emerge from this perspective. For instance, what function should the brain perform in a certain state of the external world? A mere computational model of the brain is capable of telling only how such function can be implemented in a neuronal network, provided the description of a corresponding algorithm is available. According to the theory of algorithms, any computable (i.e. describable by an algorithm) function can be performed by an infinite number of different algorithms (Cutland, 1980). Which one of them should be chosen? Why should the system perform this particular function? What factors determine the physical design of the brain? The 'Universal Computer' model in itself apparently has no room for answers to such questions.

From a different perspective, the theory of algorithms and recursive functions can help to see a similarity between objects of different nature. For example, consider language and arm movement. On the first sight, they seem to be completely different in their nature, which might imply a great difference in the mechanisms and processes of their learning. However, the inspection of mathematical descriptions of these objects reveals a great deal of similarity. A language grammar, i.e. the set of rules describing the set of syntactically correct sentences, is usually described as a set of recursive equations. A system of differential (i.e. also recursive) equations describing the arm as a mechanical system is, in fact, the 'grammar' of arm movements. For example, one can think of a set of sentences describing a certain thought and a set of arm trajectories associated with the attainment of a certain goal. If one adds a

measure of quality defined on such a set, a problem of finding an *optimal* member of the set (optimal sentence, optimal trajectory) can be posed. This would be a large step in the direction of the optimal control theory.

One easily can notice that the concepts of a universal computer and optimal control system complement each other. According to the former, a system's behavior is completely defined by providing a program that describes an algorithm for computing the system's function based on a universal computer. However, the question 'Why should the system perform this particular function?' is irrelevant in this case. In the second concept's framework, a system's behavior is defined as a solution to the optimal control problem by specifying the description of the controlled object including the criterion of optimality. Thus, the concept of optimal control provides a causal link between the properties of the environment and the system's behavior, although in general it does not give a constructive way of implementing the system's function.

3. The Idea of a Universal Learning System

It apparently is desirable for a GTL to possess the above-mentioned features pertaining to the concepts of algorithm, universal computer, and optimal control. Since the theory of algorithms constitutes the meta-mathematical core of so-called *constructive mathematics*, a combination of the above two ideas can be presented in the form of a *constructive control system*, that is a control system whose functioning can be described by an algorithm. Formal mathematical definitions of a constructive system in general and a constructive optimal control system are given in Appendix.

This paper introduces a new mathematical concept, the concept of a *Universal Learning System* (ULS), which essentially is based on the idea of constructive optimal control. Before considering its definition, note that many, if not most, mathematical constructions are finite descriptions of infinite sets. For instance, an algorithm is a finite description of an infinite in general set of pairs ⟨argument, function value⟩, a grammar is a finite description of a set of syntactically correct sentences constituting a certain language, and an optimal control system describes an optimal set of ⟨sensory input, control action⟩ pairs. Very often, they use similar constructions of higher orders. For example, a universal computer is an automaton that can perform any computable function whose algorithm's description (a program) is presented on its input; Chomsky's hypothetical generative grammar is a finite construction describing a set of grammars for the class of all possible human languages. Similarly, a ULS is defined here as *a system capable of learning to control in an optimal way any constructive system from a given class* (see Appendix for mathematical details).

Based on this definition of a ULS, one can describe a theory of learning as a theory studying the features of a ULS for different important classes of controlled systems. For instance, a neural system controlling arm movements can be described as a universal learning control system, whose class of controlled objects contains arms with different mechanical parameters determined by the mass of different muscles, the mass of fat tissue, various constraints due to a possible trauma, extra characteristics determined by objects held in the hand, etc. For a ULS as a functional model of a biological organism's interaction with the environment, the class of controlled objects may be associated with a set of objects and states of the physical environment limited by the corresponding ecological niche. In lower species, for instance, learning capabilities seem to be rather limited, which apparently reflects the relative simplicity of the corresponding class of controlled objects that can be learned, rather than contradicting the ULS model. In the course of the evolution of biological systems, one observes an increase in the complexity of the class of controlled objects and the corresponding prolongation of learning processes.

3.1. FUNCTIONAL ARCHITECTURE OF A ULS

A very simple, yet non-trivial ULS organization includes two levels of functional hierarchy (Figure 1). It corresponds to a decomposition of the controlled object into two parts: (1) variable part (or state) that changes with each time step according to a certain dynamics, and (2) constant part that is time-invariant, but differs between the members of the class of controlled objects. Note that, although no dynamics of switching between the elements

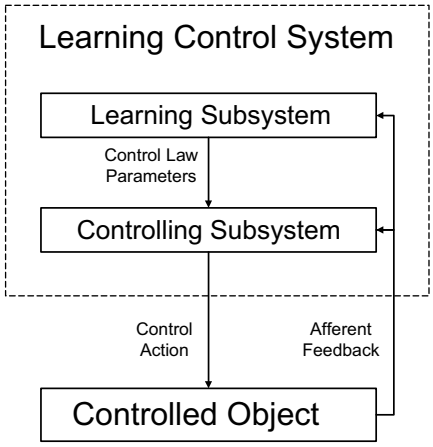

Figure 1. The hierarchy of a Universal Learning System.

of the class is specified, it is assumed that the ULS is required to be capable of re-learning after its current controlled object is substituted with a different one from the same class, thus allowing for non-stationarity of the controlled object. The ULS architecture is a result of generalization across many particular cases. For instance, in terms of the theory of automatic control, the lower level represents the control law (e.g., implemented based on an artificial neuronal net), and the higher level optimizes its parameters (e.g., synaptic weights). A detailed description of a highly efficient computer model of learning to move a mechanical object, which is based on this paradigm, is given in (Shimansky et al., 2004). In a different example, the upper level is responsible for finding an adequate set of axioms describing a scientific theory (here viewed as parameters downloaded to the lower level), and the lower subsystem's task is to perform experiments (either directly on the external reality, or the internal model of it) and test the axiomatic theory on their results. In order to do that, the lower-level subsystem encodes the experimental data in the language of the axiomatic theory and then attempts to prove or disprove the resulting statement as a theorem. Gold's paradigm of language learning (Gold, 1967) can be viewed as a particular case of this category.

After having generalized across the above two (and many other) examples, one can arrive at the following formalization. The lower level is a *universal computing* device (e.g., a universal Turing machine) that performs a so-called universal function (one of the central concepts in the theory of recursive functions) $U(C, X)$, where C is the code of a computer program and X is input data. The task of the upper subsystem in the above architecture of ULS is to find C_{opt} that is optimal with respect to a certain criterion. It appears that, in general, there is no algorithm for solving this task. In order to prove this, a certain level of mathematical formalization is required. For the reader's convenience, the formal constructions are taken outside of the main text and presented in Appendix A. The most important conclusions from the theorems that are proved there are recapitulated in an informal way below.

3.2. PROOF OF NON-COMPUTABILITY OF ULS FOR SUFFICIENTLY COMPLEX CONTROLLED OBJECT CLASSES

A formalized mathematical analysis of necessary and sufficient conditions for the existence of a ULS for a certain, relatively simple, class of controlled objects is presented in Appendix. It is based on the following idea. A general function that can be described by a computer program (known as *partial recursive function*, PRF) may not be defined on certain values of its arguments, indicating that the process of computation will never be finished on such inputs. In order to make our mathematical model adequate with regard

to the physical reality, it must be required that the behavior of each system from the class of controlled objects for a given ULS be described by a *completely defined* computable function (*total recursive function*, TRF).

It is proved (Theorem 1) that a ULS in general must be capable of enumerating the set of TRFs corresponding to the given class of controlled objects. The proof is based on constructing the criterion of control optimality so that the ULS has to create a functionally exact internal model of the controlled object. It is well known that the enumeration of a set of TRFs in general may not be computable (i.e. it may not be possible to describe the enumeration procedure by an algorithm). For instance, the set of all TRFs is not algorithmically enumerable (Cutland, 1980). Hence, the behavior of a ULS in general *cannot be described by an algorithm*.

The possibility to enumerate a given class of computable functions appears to be not only necessary, but also sufficient for the existence of a ULS for a given set of controlled systems (Theorem 2). In classical mathematics, this allows one to prove the existence of a ULS for any class of controlled systems regardless of whether it is algorithmically enumerable or not. This proof (Theorem 3) is based on an application of the so-called *axiom of choice*, which asserts the existence of a possibility to choose an element from any given set (since the set of TRFs is countable, only the countable version of the axiom of choice is needed). This axiom plays a truly unique role in mathematics by allowing proof of the existence of a mathematical object without providing an algorithm for its construction. Such an algorithm is required in constructive mathematics but not in classical mathematics. It immediately follows from the axiom of choice that there is a way (though not necessarily an algorithmic procedure) to enumerate any given set. Although this provides a possibility to prove the *existence* of a ULS for any given class of controlled systems, it does not help one to understand *how to create* a ULS for a class that is not algorithmically enumerable.

It is important to note that the set of all PRFs (including all TRFs) *can* be enumerated by an algorithm. However, there is no contradiction between this fact and the absence of an algorithm for enumerating all TRFs. It is possible to create a list of PRF codes, but it is impossible to distinguish (algorithmically) between TRF and PRF codes. This is very similar to being able to make a list of all finite sequences of printable characters when looking for an answer to a certain question, without knowing how to select a sequence representing the right answer, or how to select a sequence that would make any sense at all.

In the theory of recursive functions, the notion of an *oracle*, a hypothetical external device utilized for performing an extra function as an elementary operation, plays a very important role. A so-called *halting oracle* can be used for determining whether a given computer program will stop on a given input. If available, it provides a possibility to distinguish between PRFs and

TRFs in a list of all PRFs, and, consequently, enumerate TRFs. This possibility might seem to have only pure theoretical importance. However, as will be mentioned below (Section 5), it appears feasible, at least in principle, to implement a quantum-mechanical halting oracle and, consequently, a non-computable ULS.

The above approach to defining the paradigm of universal learning and a ULS is not the only possible. Hutter (2003) defines a ULS (Universal Artificial Intelligence Model in his terminology) as an intelligent agent whose goal is to maximize future rewards in a probabilistic environment that consists of a certain (but unknown a *priori*) computable stationary probability distribution over a set of deterministic environments (i.e. a set of all PRFs). The main problem addressed in that study is to determine the actual probability distribution. A simple and elegant solution of this problem is based on Solomonoff s theory of universal induction, which combines Kolmogorov's measure of algorithmic complexity and Bayesian prediction. At the same time, the assumption of stationarity of the above probability distribution and, therefore, the entire stochastic environment can be viewed as imposing serious limitation on the paradigm's applicability to modeling adaptation of biological systems to environmental changes. Hutter's model of ULS appears to be also non-computable, which is fully consistent with the proof described in the previous section. He also describes a computable version of the AI model, which is the best among those bounded by given complexity and time limits.

4. GTL as a Theory of ULS

A general theory of learning (GTL) has to be more complex than just an algorithmic theory to qualify as a counter-example to argue against Chomsky's denial of a possibility to create a non-trivial theory of learning. The above-described results of the mathematical treatment of this issue unequivocally demonstrate that an algorithmic approach is not sufficient for developing a GTL. The establishment of the non-trivial character of a GTL as a general theory of Universal Learning Systems constitutes the main result of this study. It appears that the functioning of a ULS for a sufficiently wide class of controlled systems *cannot* be described by an algorithm, indicating that a GTL has to be essentially non-algorithmic.

4.1. ULS THEORY AND NATURE/NURTURE ISSUE

Let us recall once more nativists' scheme of the development of a biological organism. Any new feature (compared to the organism's progenitor) is

created at the time of genome mutation and recombination. Then the genetic program starts to function, and the ontogenesis essentially is the processing of information obtained from a specific environment by the genetic program. In different terms, the epigenetic rules automatically 'fill out' the innate matrix with data received from the environment. Thus, the only 'truly creative', non-algorithmic aspect in the process of the creation of a new biological organism is the mechanism of random modification in the genes. Is this mechanism creative from a mathematical perspective, however?

It is well known that probabilistic algorithms (such algorithms include calls to a random number generator) do not extend the class of computable functions (Leeuw et al., 1956). This is so primarily because one can substitute a random number generator with a deterministic algorithm for enumerating finite sequences of '0'-s and '1'-s, provided the parameters of the required probability distribution (e.g., the probability of '1') are specified as computable numbers. Randomness helps when the enumeration of *sequences* of objects rather than objects themselves is required. As an alternative to randomness, one would have to have an infinite memory not to repeat the same sequence infinitely many times. Thus, the utilization of a random number generator as an *oracle* (i.e. an external device utilized for performing an extra function) instead of a sophisticated deterministic algorithm greatly simplifies the enumeration task. However, non-deterministic algorithms do *not* provide any extra creative power.

This well-known mathematical result may seem somewhat unexpected to a biologist, since it usually seems intuitively 'obvious' that randomness plays a critical role in learning. For instance, it has been proposed that novel properties of biological systems can emerge from the random recombination of elementary building blocks (Cziko, 1995). The creative power of this mechanism, however, does not reach beyond the domain of computable functions, since all possible DNA sequences (as all finite sequences of characters from a finite alphabet) clearly can be enumerated by a relatively simple algorithm. In different terms, the process of gene modification itself can be viewed as just 'filling out' the corresponding set of DNA sequences as a 'matrix'.

Let us recall now that Chomsky ascribed creativity to the mechanism of gene mutation not because of some positive reason, but rather as a last resort, just since it 'cannot be anywhere else' (see Section 1). It is clear from the above analysis that if random changes in the network of billions of neurons in the brain do not provide creative power, then such modifications in incomparably simpler structure of DNA sequences would so much the less. Thus, one has to conclude that the nativists' model of the development of biological organisms is wrong. The nature's creativity in the development of biological organisms cannot be reduced to random changes in their genome. At the same time, *only non-innate mechanisms* can be responsible for the non-algorithmic characteristics of an organism's development.

Hence, we are left with two main alternatives with regard to the model of the development of biological systems. Namely, either there is no creation (as a non-algorithmic process) at all, or the process of creation affects the entire biological system at any stage of its development. Before going any further, it should be noticed that this philosophical dilemma is of a much greater scale and much older than the nature/nurture issue. A few more steps along this direction will be made below with the main purpose to demonstrate that the conceptual basis of ULS is adequate for dealing with this problem. According to the first alternative, everything has been created by a Mega-Algorithm of tremendous complexity, and we do not perceive it as such just because the complexity of the algorithm of our brains' functioning is much smaller. In this case, all creative power has been already used for creating the Mega-Algorithm (or one could assume that it always existed). Second, acts of (non-algorithmic) creation do occur, at least once in a while. As Niels Bohr (a Nobel laureate in physics, one of the fathers of quantum mechanics), mentioned, physics is concerned with how we can *describe* the reality, rather than what the reality is. It is very important to acknowledge that, given a certain complexity of the model of physical reality in the brain of the investigator, the investigator would not be able to distinguish between the case in which there is a Mega-Algorithm complexity of which is greater than that of the algorithm of brain functioning (and consequently the model's complexity) and the alternative situation in which the reality cannot be fully described by an algorithm in principle. In both cases, however, an increase in the complexity of the algorithm of a brain's functioning apparently is possible!

It should be emphasized that, based on our current knowledge about the physical reality, the above two models are not equal in their likelihood. There are two facts: (1) the mathematical foundation of modern physical theory requires the axiom of choice (or its analog) and (2) this axiom allows the enumeration of sets that are not algorithmically enumerable, thus permitting the existence of a non-algorithmic ULS. Therefore, the 'true creation' alternative appears to be in much better correspondence with the general model of our universe created so far (note that the term 'created' here relates to both the 'model' and 'universe'!). Some other important arguments in favor of developing a theory of non-algorithmic ULS (in addition to that of algorithmic ones) will be presented farther below.

4.2. STRUCTURE OF ULS THEORY

It has been demonstrated that a non-trivial GTL can be developed in the form of a ULS theory. The term 'general' here by no means is intended to indicate that the only goal is to find the 'largest common denominator' for

the adaptation of a primitive unicellular organism to its surroundings and development of scientific theories. The goal is to create a scalable theory that would help to get into *details* of each adaptive process. The general structure of this theory can be presented as a tree. The root node should include features common to ULS corresponding to different classes of control objects. One of the main such functional features, for instance, is the concept of internal model of the controlled object (Baev and Shimansky, 1992; Gerdes and Happee, 1994; Wolpert et al., 1995; Shimansky, 2000). An internal model itself also can be presented as a ULS, where the lower level is responsible for the identification of the controlled object's current state, and the upper level, for the identification of the IM's parameters. Special parts of the root stratum of the ULS theory should be allocated to the classification of internal model types, functional decomposition of a complex internal model, interaction between the internal models and other functional components, etc. Every important class of controlled objects (such as devices without internal memory, finite automata, systems with finite-dimensional continuous state space stack automata, etc.) requires a special subdivision (i.e. a 'node') of the ULS theory. Each node includes theoretical concepts common to the corresponding controlled object subclass and further classification. Needless to say that the tree will grow as theory development progresses, and a significant amount of interaction between theoretical directions corresponding to different nodes should be expected.

The results of this study emphasize the importance of the problem of enumerating objects from a certain class. Computationally, this problem is of the most difficult type. It may not seems so urgent for relatively simple classes objects of which have the same complexity and can be parameterized by a finite set of variables. The problem of enumerating such objects *optimally*, so that the corresponding ULS learns fast enough, apparently is the most critical (a typical example is an adaptive system for movement control). A different situation exists in those cases in which controlled object complexity significantly varies between different elements of the class (e.g., in such tasks as orientation in an extra-personal space and language learning). In that situation, the mere problem of enumerating the objects becomes computationally challenging enough to appear in the forefront. A complex controlled system usually can be presented as a combination of simpler subsystems, which allows one to organize the enumeration of a class of such controlled objects based on enumerating the subsystem classes and the patterns of their combination. Similarly, the control of a complex controlled object involves controlling each subsystem. Hence, development of a complex ULS should be based on simpler ULS as subsystems. The possibility of performing a functional decomposition of a complex learning system into simpler subsystems of the same general type is an important advantage of the ULS framework.

4.2.1. *Algorithmic ULS*

The recent progress in computer technology makes of high practical importance the development of ULS that can be implemented based on a computer. In fact, all known models of learning mechanisms are algorithmic. The ULS approach is not an alternative to existing learning-related paradigms, such as machine learning methods, unsupervised learning algorithms, error back-propagation networks, reinforcement learning approach, etc. It is supposed to be developed as a unifying basis for integrating the existing methods and concepts in order to make them available and useful for experimental research. Regrettably, because of paper size limitation, an analysis required to demonstrate that the ULS approach indeed can provide such a constructive basis cannot be included here. At the same time, a theory of computational ULS apparently should be developed much further than just a basis for existing theories of learning. It can be demonstrated that based on the concept of so-called constructive object, which is no less fundamental than the notion of an algorithm, but so far much less well known (Uspensky and Semenov, 1993), one can develop a theory of learning systems with complex, hierarchical parameter structures and multiple optimization criteria with their own hierarchy.

4.2.2. *Non-Algorithmic ULS*

It has been shown that a non-algorithmic ULS is a valid mathematical object. Moreover, the possibility of its existence does not seem to contradict any known physical law. Hence, a GTL can be developed as a theory of ULS. However, this possibility by itself does not necessarily mean that such a theory *should* be developed. Indeed, it would be quite a peculiar entity, since by definition it cannot be constructive in the usual sense of algorithmic constructiveness. Is there a more general type of constructiveness than algorithmic? So far, we only know that all attempts to mathematically formalize a system capable of performing an effective procedure led to a schema equivalent to Turing machine (i.e. to a Universal Computer). At the same time, we know that an algorithmic procedure is not sufficient in general for describing the reality. This awareness began at the beginning of 20th century from Gödel's theorem that effectively put an end to Hilbert's program of fully automating mathematical proofs. From another perspective, classical mathematical analysis, the foundation of contemporary theoretical physics, requires the axiom of choice (at least for countable sets). Perhaps few scientists would try to deny the importance of working on the development of a non-algorithmic GTL from the perspective of fundamental research. Nonetheless, would such a theory have any practical value? Before trying to answer this question, it is important to clarify the advantage of using a more computationally powerful device in general. Consider the following two examples.

The first example is the calculation of the factorial function of an integer number $f(n) = 1 \cdot 2 \cdot 3 \cdots (n-1)n$. To calculate this function with a device without internal memory based on a feed-forward computational scheme, one has to provide a direct calculation formula. However, the length of an instruction for factorial computation based on a direct formula is proportional to the value of the function argument. Writing and transmitting such instruction for a large n each time one wants to compute the function constitutes a problem by itself. If the computing device is of a finite automaton (FA) type (i.e. with a finite amount of internal memory), a much more concise and elegant way of describing computation of the factorial function can be used. It is based on recursion: $f(0) = 1$, $f(n+1) = (n+1)f(n)$. This description's length is invariant with relation to n (here we do not consider the length of the representation of n itself). This instruction needs to be sent to a computer only once before computing the function for a set of argument values.

The second example is the task of navigating through a maze. For practical reasons, we may consider the maze complexity limited, so that a FA of sufficiently large state space could be used. It could be easily shown that the description of the corresponding schema of state transitions for the FA would be enormous, even for classes of relatively small mazes. With a universal computer, however, a program that solves the same task based on the stack technique (which requires dynamic memory allocation) can be incomparably smaller.

The above examples illustrate the general fact that a more computationally powerful device allows a more concise coding of knowledge. A universal computer is the most powerful automaton, meaning that it can perform any algorithmic procedure. A ULS in general is required to have more than algorithmic power. Making an inductive inference (Figure 2), one can expect the ULS theory to ultimately provide a still more powerful way of encoding knowledge.

5. Non-Computable ULS as a Physical System

The fact that the idea of a non-algorithmic ULS is consistent with classical mathematics is a strong indication of a possibility that such a ULS exists as a physical system. It follows from the above analysis that the computational properties of physical systems are not sufficient in general for constructing a ULS, but it is not immediately clear which other physical properties could be used for that. Whatever those properties are, they obviously cannot be adequately described based on a computer model. Strong evidence that such properties exist comes from the theory of quantum mechanics. First, quantum mechanics is inconsistent with a local hidden-variable model (Bell, 1966).

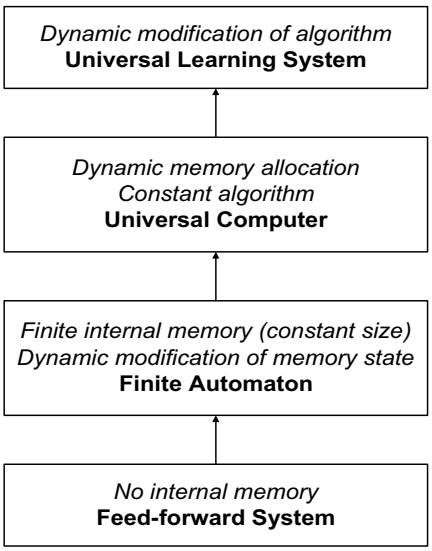

Figure 2. Inductive series of devices with increasing computational power.

Stated differently, quantum mechanical systems are of non-deterministic nature not because an external observer lacks information about it, but rather because any deterministic model with hidden parameters, in which physical systems are not allowed to interact with speeds higher that the speed of light, is inadequate. Since the code of a computer program can be considered a generalization of a hidden variable's value, the above principle indicates that quantum-mechanical systems in general cannot be modeled by an algorithmic procedure. Second, it appears that there exist a quantum-mechanical solution to the tenth problem of Hilbert (finding integer roots of diophantine equations in finite number of steps), which is known to be equivalent to the non-computable problem of determining whether a given computer program will halt on a given input (Kieu, 2003). Stated differently, current theory of quantum mechanics is consistent with the possibility of creating a quantum-mechanical halting oracle, which, as it was mentioned above, can solve the problem of TRF enumeration.

Since quantum mechanics is the most precise physical theory to date (Lawrie, 1990), the above theoretical observations strongly indicate that non-algorithmic ULSs exist in the form of a physical system. One may hypothesize, for example, that the brain is such a system. Furthermore, one can intuitively expect that the most powerful learning functions of the brain be based on non-algorithmic properties of it as a physical system. In this regard, an important question arises, 'How such properties can help increase the learning power of a ULS?' Some possible approaches to finding an answer to this question are outlined below.

5.1. SIMPLE SOLUTION OF NON-STOPPING PROBLEM

In order to understand better the 'game rules' in the physical world, let us look at a possible interaction between a physical control system and a physical controlled object. For concreteness, we can assume that the control system design is based on a universal electronic computer, since the latter is a physical realization of a Turing machine, and the controlled object is just another physical system. The computer cannot enumerate all and only TRFs, but it can enumerate PRFs by enumerating algorithm descriptions, i.e. programs. In this case, however, a program generated may work without stopping after having received an input from the controlled object. In physical reality, it takes non-zero time to perform an elementary step. Since the computer is not supposed to interact with the controlled object until the computation of control action is finished, it may never resume its interaction with the controlled object. From another perspective, the absence of control signals in general is just one of the logistically possible control actions, since the controlled object does not have to wait for a signal from the control system. Therefore, the controlled object would continue moving along the 'ballistic' phase trajectory, which obviously may be far from being optimal. Thus, not only the control action should be computed in a finite number of steps, but also the computation time must be small enough to satisfy time constraints imposed by the controlled object dynamics.

This complication, however, is accompanied with an important benefit. A physical controlled object would not wait for the control system to finish its computations, but for the same reason it can interrupt the computational process by sending a special signal to the control system. That signal can force the computer to generate the next PRF and start performing it. If an interruption signal is sent every time and only when the state of the controlled object is 'undesirable', the problem of non-stopping can be eliminated. However, the price of it appears to be quite substantial: now the control system must learn not only to compute optimal control actions, but also to finish the computation in time, i.e. before the state of the controlled object becomes 'undesirable' and an interruption is initiated. There are infinitely many programs for the computation of a given TRF. If a program exists that can compute an optimal TRF within the time limits imposed by the controlled object behavioral properties, it can be found by the above enumeration mechanism in the finite number of steps. This is so because there are a finite number of items in the list of all PRFs until an optimal TRF is generated, and it takes a finite number of steps between generations of two successive PRFs.

It looks like the above method eliminated the problem of internal inability to determine whether a given program will stop on a certain input. The problem is not resolved, though; it rather is bypassed by introducing *a new type of feedback signal*, namely, process interruption. Stated differently, the

ULS is relieved from the task of making decision when to stop computation; the task is transferred to the environment, and the ULS now just passively complies with the decision. Note that in general it is not possible to predict the occurrence of the interruption signal based on an algorithm. Otherwise, one could combine that algorithm with the procedure of PRF enumeration to create an algorithmic ULS that would not need any interruption signal.

5.2. ENUMERATION OF PHYSICAL DESIGNS

It may seem on the first sight that we did obtain a *constructive* enumeration of TRFs. However, by having opened the Pandora's box of the physical world, we let out many other problems. In order to find a control system whose response time fits the above constraints, enumeration of computer programs is not sufficient; it is also necessary to enumerate *physical designs* of the universal computer, because the computation time substantially depends on it. The history of computers shows how big the range of serial computer speed is depending on the physical implementation of the processor, memory and other parts. One can expect that the speed of parallel computers and computational networks depend even more dramatically on their physical design. Furthermore, when dealing with the problem of physical design enumeration, one should consider the design of the information interface. An enhancement of the input part (e.g., increase in the speed of information transfer, creation of new channels) can increase the observability of the controlled object and consequently the speed of the system's performance. Similarly, a development of the output effector part can increase the controlled object controllability, which may contribute even more to the overall efficacy of the entire control system. Such changes in a control system are hardly ever considered as a part of the learning process. They can be quite important, though, because a lack in observability or controllability of the controlled object due to an underdevelopment of the control system's interface may not be compensable by any increase in the computational power of the 'thinking' part. Thus, the process of physical design enumeration must include all the parts of the control system.

There are at least two main different methods for modifying a system's physical design. First method consists in switching between elements of a certain pre-determined set of alternatives expressed in terms of the system's structure and parameters. A computational model of physical design enumeration according to this method apparently can be developed. The second method is characterized by permitting modifications that affect not only the system's motion along its existing degrees of freedom, but alter these degrees, including the creation of new ones. This can be done in practice by simply infusing a certain amount of energy into the system, which is another example of feedback type that cannot be used, if only computational properties of

physical systems are considered. If the hypothesis that physical processes in general are non-computable is correct, enumeration of a non-recursive set of physical designs can be achieved based on the second method. This apparently can bring greater learning power at the expense of a possibility of altering the system to such an extent where it no longer would be capable of performing its function, which, for a biological organism, would mean its death. It is conceivable that there are many different types of non-algorithmic modifications of physical design; consequently, a compromise between the capacity to learn and functional stability has to be found for each ULS.

6. Conclusions

It has been demonstrated that neither computational properties of physical systems, nor random mutations in the genome of a biological organism can account for the creativity that manifests itself in acquiring new skills and developing scientific theories. This conclusion conforms to the intuitive idea that learning function of the brain, the most complex known physical system, is essentially more than an innate program for processing sensory information. There are strong indications that the main source of the creative power of learning is related to physical interactions of a ULS with its environment. However, it is not yet clear what other physical properties are critically important for creating a ULS. This problem apparently should constitute a substantial part of GTL.

Perhaps the main conclusion from this study is that a non-trivial and rigorous GTL can be developed essentially as a theory of ULS. It cannot be a pure mathematical theory, for it has to include non-computable aspects of physical reality. Therefore, it needs to be developed essentially as a physical theory. Its subject, unlike that of any other theory, includes formalization of theory creation in general, which makes it applicable to itself, thus constituting a significant element of self-awareness. This unique feature should make GTL attractive to increasing number of scientists who strive to formalize the idea of consciousness. The ULS approach opens an entirely new perspective on creating a theory of brain functioning and the processes of biological systems evolvement. It becomes clear that the development of such a theory will require collaboration between biologists, physicists, and mathematicians on a new level.

Acknowledgements

I am grateful to Vladimir Nevzorov and Marcus Hutter for valuable comments and suggestions, and to Eric Schechter for the very helpful discussion on the relationship between constructive mathematics and physics.

Appendix A

The material given below serves a double purpose. First, it provides mathematical formalizations for the main ideas informally presented in the main part of the paper. Second, it outlines a few initial steps along the development of a general mathematical theory of learning systems. It includes formal definitions of a constructive system and a Universal Learning System (ULS), necessary lemmas and three theorems with proofs that describe necessary and sufficient conditions for ULS existence. The definitions and proofs are based on the notions and ideas of the theory of algorithms and recursive functions. A simple and clear description of the latter can be found in Cutland (1980). A good overview of the fundamental concepts and the main achievements of this theory are given in Uspensky and Semenov (1993). All the theoretical results that are referred to below without giving a source can be found in either book. The main terms are described below as a brief reminder for the reader's convenience.

We shall use the standard symbol N to denote the set of natural numbers $\{0, 1, 2, \ldots\}$, and, as usual, N^n denotes the set of ordered n-tuples $\{(x_1, \ldots, x_n), x_1 \in \mathrm{N}, \ldots, x_n \in \mathrm{N}\}$. The set of all finite sequences of natural numbers is denoted as N* (for any set **A**, **A*** designates the set of all finite sequences of **A** elements). For all the functions we will be dealing with below, the domain is a subset of N^n and the range is a subset of N (short notation: $\mathrm{Dom}(f) \subseteq \mathrm{N}^n; \mathrm{Ran}(f) \subseteq \mathrm{N}$). A function from N^n to N domain of which is known to be the whole of N^n ($\mathrm{Dom}(f) = \mathrm{N}^n$) is called a *total function*, otherwise (when it is not known whether the function is total or not, or when it is known that the function is not total) it is called a *partial function*. Therefore, the set of total functions is a subset of the set of partial functions. If an algorithm exists for calculating a given function, the function is usually described in such terms as *effectively calculable*, or *computable*. Computable functions are also called *recursive functions*. An algorithm for computing a certain partial recursive function (PRF) will stop in a finite number of steps on all and only those inputs that belong to its domain, and will not stop on any other input. Thus, an algorithm for a total recursive function (TRF) will stop on any input.

For a given set $\mathbf{A} \subseteq \mathrm{N}$ a function $q_{\mathrm{A}}(x)$ defined as $q_{\mathrm{A}}(x) = 1$, if $x \in \mathbf{A}$, otherwise $q_{\mathrm{A}}(x) = 0$ is called the characteristic function of the set **A**. If that function is computable, the set **A** is called a *recursive set*. A set $\mathbf{A} \subseteq \mathrm{N}$ is called *recursively enumerable* (or just *enumerable*) if there exists a computable function $g_{\mathrm{A}}(x)$ defined as $q_{\mathrm{A}}(x) = 1$, if $x \in \mathbf{A}$, otherwise $q_{\mathrm{A}}(x)$ is undefined. Any recursively enumerable set is a domain of a partial recursive function. At the same time, it is a range of a total recursive function (which can be used to generate the set's elements).

The set of all computable functions (i.e. all PRFs) can be generated by a certain calculus that includes three simple basic functions (zero function,

successor function, and projection function) and three *operators* (functions whose arguments and values are functions) of *substitution, recursion, and minimalisation*.

A description of an algorithm for computing a certain PRF on a certain automaton is called a *program*. Programs (and therefore algorithms and corresponding PRFs) can be effectively encoded by natural numbers. There exists a universal algorithm that takes such a code on its input, decodes it, and then works according to the resulting description. In terms of recursive functions, one of the corresponding universal function's arguments is a description of a particular recursive function. Universal computers (including the universal Turing machine) are actually automatons that work according to a universal algorithm.

A1. MAIN DEFINITIONS

In order to describe the setup of the problem of optimal control and the process of learning to control in an optimal way, it is necessary to define first the notions of *system* and *control process*. A good description of the modern theory of optimal control can be found in Bertsekas (2000). A universal learning problem setup that is very similar to the one presented below is described in Hutter (2001).

System. A *system* is defined here as a functional aggregate that includes *input, output, internal memory*, and *processor*. A variable and a certain set of its possible values are associated with each one of the first three parts: input $X \in \mathbf{X} = \mathrm{N}^n$, output $Y \in \mathbf{Y} = \mathrm{N}^n$, internal state $S \in \mathbf{S} = \mathrm{N}^*$. The processor performs the system's function $f : \mathbf{X} \times \mathbf{S} \to \mathbf{Y} \times \mathbf{S}$, that can be presented as a composition of two functions f^{Y} and $F^{\mathrm{S}} : S = f^{\mathrm{S}}(X, S)$, $Y = f^{\mathrm{Y}}(X, S)$. A system works by steps. On each step i, the current values of input X_i and internal state S_i are transformed into the output value Y_{i+1} and the next internal state S_{i+1} according to the corresponding system's functions. So a system is a structure $\langle \mathbf{X}, \mathbf{S}, \mathbf{Y}, f \rangle$. It is important to emphasize that the notion of control system used here is pure functional and should not be mixed with a concept of control system as a physical entity.

There are two fundamental properties of a system as an object of control: *observability* and *controllability*.

Observability is a possibility to obtain information about the system's state based on a finite history of its output. It can be formally defined as the following. A system $\langle \mathbf{X}, \mathbf{S}, \mathbf{Y}, f \rangle$ is observable if for its every state $S_i \in \mathbf{S}$ there exist a number m and a total function h such that $S_i = h(Y_i, Y_{i+1}, \ldots, Y_{i+m})$. A system is *fully*, or *immediately observable* if and only if the smallest m is equal to zero for every state, i.e. if any state can be unambiguously determined based on the corresponding output at the same step.

Controllability is related to the freedom of moving the system in its state space by acting on its input. A system $\langle \mathbf{X}, \mathbf{S}, \mathbf{Y}, f \rangle$ is *controllable* if it can be moved from any state S_i to any state S_j by applying a finite sequence of control actions $(X_i, X_{i+1}, \ldots, X_{i+m})$ to it's input. A system is *fully controllable* if and only if the smallest m is equal to zero for every pair of its states, i.e. the system can be moved from any state to any state in one step.

Recursive system. A system is *recursive* by definition, if its functions are computable. In other words, the behavior of a recursive system can be described by an algorithm. Similarly, a system's properties are *recursive*, if the functions involved in their description are recursive.

Constructive system. An observable and controllable recursive system, whose properties are also recursive, is defined as *constructive*. Note that the functions that describe a constructive system's properties are total recursive.

The functioning of a recursive system can be emulated by a Turing machine or any other universal computational automaton. It should be noted that the internal memory is a pure functional (software) notion and should not be mixed up with a (hardware) memory like Turing machine's tape that is necessary to compute the function.

Optimal control process. In the scheme of a control process, two systems are inter-connected so that the output of each one of them is connected to the input of the other. One of those two systems is called *a control system*, and the other is called *a controlled object*. The asymmetry in naming the systems reflects mainly the difference in our viewing their functional roles in their interaction with each other, rather than an objective difference between these systems. The process of control also can be viewed as a game between two or more systems, one of which is separated out to study its optimal functioning determined by the game objective. That system is called *a control system*. The rest of the systems are viewed in combination as *a controlled object*. It is assumed below that the controlled object is a constructive system.

A control system $\boldsymbol{A} = \langle \mathbf{X}_\mathrm{A}, \mathbf{S}_\mathrm{A}, \mathbf{Y}_\mathrm{A}, f_\mathrm{A} \rangle$ and a controlled object $\boldsymbol{B} = \langle \mathbf{X}_\mathrm{B}, \mathbf{S}_\mathrm{B}, \mathbf{Y}_\mathrm{B}, f_\mathrm{B} \rangle$, where $\mathbf{X}_\mathrm{A} = \mathbf{Y}_\mathrm{B} = \mathbf{X}$, $\mathbf{X}_\mathrm{B} = \mathbf{Y}_\mathrm{A} = \mathbf{U}$, interact with each other in the following way. Let us designate by i_A and i_B step numbers in $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively. At each step $i = i_\mathrm{B}$ of the control process, the control system receives information $X_i(i_\mathrm{A} = i)$ from the controlled object and computes a control action $U_i = U_{i_A+1} = f_\mathrm{A}^\mathrm{Y}(X_i, S_{\mathrm{A}i})$ and a new internal state $S_{i_\mathrm{A}+1} = f_\mathrm{A}^\mathrm{S}(X_i, S_{\mathrm{A}i})$. Then a new output $X_{i+1} = X_{i_\mathrm{B}+1} = f_\mathrm{B}^\mathrm{Y}(U_i, S_{\mathrm{B}i})$ and a new internal state $S_{i_\mathrm{B}+1} = f_\mathrm{B}^\mathrm{S}(U_i, S_{\mathrm{B}i})$ are computed in the controlled object, and the step numbers are updated: $i_\mathrm{B} := i_\mathrm{B} + 1$, $i_\mathrm{A} = i_\mathrm{A} + 1$ and $i := i + 1$. For an adequacy of this mathematical model, it is required that controlled objects will not work without stopping, therefore it is assumed that the functions f_A^S and f_B^S are total in every controlled object.

The criterion of control *optimally* is based on an estimate that determines the extent of 'undesirability' of controlled object state for the control system.

Such estimate can be introduced in different ways. Here the set of criterial values $\mathbf{E} = \{0, 1\}$ is defined as a one-dimensional subspace of $\mathbf{X}$. So the latter can be presented as a Cartesian product: $\mathbf{X} = \mathbf{Z} \times \mathbf{E}$, where $\mathbf{Z}$ and $\mathbf{E}$ are the informational and criterial parts of the control system's input, respectively. Control *optimization* is defined for a step interval $[i, m]$ of the control process as a minimization of the criterial sum $\sum_i^m = \sum_{j=1}^m e_j$, where $e_j \in \mathbf{E}$. Having introduced $\sum_i = \lim_{m\to\infty} \sum_i^m$ and defined that $\sum_i < \sum_i'$ if and only if $\exists M \forall m > M(\sum_i^m < \sum_i'^m)$, where $\sum_i^m$ and $\sum_i'^m$ correspond to different instances of the control process, we can expand the definition of the optimality criterion by including the cases of infinite step interval and divergence of the criterial sum. A function performed by the control system is *optimal* if it minimizes the above criterion. For the purposes of the analysis presented below, we can assume that there always exists a total computable control function such that the criterial value e_i becomes zero after finite number of steps, which ensures finiteness of the corresponding criterial sum, thus eliminating complications arising in connection with the general case of infinite-horizon problem (Bertsekas, 2000).

Learning system. Here a *learning system* is defined as a two-level hierarchical construction (Figure 2). The lower-level system (controlling) subsystem $\boldsymbol{C} = \langle \mathbf{X}_\mathrm{C}, \mathbf{S}_\mathrm{C}, \mathbf{Y}_\mathrm{C}, f_\mathrm{C} \rangle$ directly controls an object $\boldsymbol{A} = \langle \mathbf{X}_\mathrm{A}, \mathbf{S}_\mathrm{A}, \mathbf{Y}_\mathrm{A}, f_\mathrm{A} \rangle$, and receives input information from both the controlled object $\boldsymbol{A}$ and the upper level system (learning subsystem) $L = \langle \mathbf{X}_\mathrm{L}, \mathbf{S}_\mathrm{L}, \mathbf{Y}_\mathrm{L}, f_\mathrm{L} \rangle$ which, in turn, receives input from the controlled object $\boldsymbol{A}$ and sends control actions to the lower subsystem $\boldsymbol{C}$. According to the described connections, $\mathbf{X}_\mathrm{L} = \mathbf{Y}_\mathrm{A} = \mathbf{X}$, $\mathbf{X}_\mathrm{C} = \mathbf{X} \times \mathbf{Y}_\mathrm{L}$, $\mathbf{X}_\mathrm{A} = \mathbf{Y}_\mathrm{C} = \mathbf{U}$. The learning subsystem's output $Y_\mathrm{L} \in \mathbf{Y}_\mathrm{L}$ can be viewed as a set of parameters $P = f_\mathrm{L}(\mathrm{X}, \mathrm{S}_\mathrm{L})$ downloaded to the controlling subsystem. The theory of recursive functions offers a natural generalization of P: the function f_C is a *universal* function that treats P as a *code* of a particular function to be performed by the controlling subsystem. Thus, the learning subsystem's role is to find a code of an optimal control function and download it to the controlling subsystem that directly acts on the controlled object. From another perspective, the learning subsystem computes a function $P = L(\alpha_X)$, where $\alpha_X \in \mathbf{X}^*$ is a finite sequence of the controlled object's output values. This function will be referred to below as to a *learning* function.

This definition by no means pretends to be the most complete definition of learning system; nevertheless, it is sufficient for the learning problem analysis given below.

Universal Learning System (ULS). This is the central notion of the theory of learning. A ULS is defined here as a learning system that, for any object from a certain class, is capable of finding an optimal strategy of controlling the object in finite number of steps after having been *connected* to that object. The operation of *connecting* a learning system to an object should not be viewed as an establishing of a physical link. It consists of assigning a new

value to the variable 'controlled object' and initializing the internal memory of the learning system.

It is extremely important to note that the function codes generated by the learning subsystem of a ULS and downloaded to its controlling subsystem must be the codes of *total functions only*. Otherwise, the whole system might work without stopping on a certain input from the controlled object, which contradicts the above definition of a ULS. Note also that this definition of a ULS is not specific to any particular class of controlled objects.

A2. THE PROBLEM OF ULS CREATION

Before analyzing the problem of learning to control an object in an optimal way, let us first take a look at a simple particular case of the optimal control problem, the problem of finding an argument corresponding to the global minimum of a total function $f: x_{\min} = \operatorname{argmin}_x[f(x)]$, assuming it has one. The function f is computed by an external device, i.e. $f(x)$ is obtained on its output after having sent x to its input. If the global minimum value $f_{\min}$ is known *a priori*, the problem can be solved in a finite number of steps based on the operator of minimalisation: $X_{\min} = \mu x[f(x) = f_{\min}]$. This notation corresponds to the following algorithm: enumerate x values starting from zero until the first (i.e. the smallest) one that satisfies the equation shown in brackets is found. If the global minimum is just known to be finite, but its exact value $f_{\min}$ is not known *a priori*, $x_{\min}$ and $f_{\min}$ still can be found in a finite number of steps by the following simple algorithm: set $x_{\min} = x = 0$, $f_{\min} = f(x)$, update $x_{\min}$ and $f_{\min}$, while enumerating x, each time when $f(x) < f_{\min}$. However, there is no criterion for stopping in this case: there is no guarantee that a value still less than the current $f_{\min}$ cannot be found by continuing the search no matter for how many steps in a row $f_{\min}$ has not been updated.

For the domain of multi-argument functions, the argmin function can serve as an operator: $g(Y) = \operatorname{argmin}_C[f(C, Y)]$, where C is a code for the combination of optimized parameters. In a general case, C is a code of a computer program that takes Y on its input. Here we assume that the operator application does not lead outside of the class of total functions. This operator can be combined with other operators defined on recursive functions. We assume here that the control optimization problem has only one criterion; therefore, only one instance of argmin operator is needed.

The absence of a criterion for stopping is one of the most general features of a learning process. In a case in which the problem of ULS existence for the class of all recursive objects is considered, the existence of such criterion would mean a possibility to fully identify any controlled object by a finite history of its behavior. In terms of the recursive function theory, this means

identifying any recursive function by a finite set of argument-value pairs, which is obviously impossible. Thus, in the most general case, we cannot require that a ULS is capable of stopping in a finite number of steps after the best solution has been found. We can only require that it *seek* a better solution and be capable of finding the best one in a finite number of steps, if it exists. Stated differently, an adequate requirement for a ULS is that it must be capable of *continuing the process* of learning. Note that the problem of optimization speed is not addressed here.

A3. NECESSITY OF THE ENUMERATION OF TOTAL RECURSIVE FUNCTIONS FOR LEARNING UNIVERSALITY

The main purpose of this section is to show that there exist classes of controlled objects that require a non-constructive ULS. One such class, $\mathbf{R}_0$, is constructed below. For every controlled object $\boldsymbol{A} = \langle \mathbf{U}, \mathbf{S}, \mathbf{X} = \mathbf{Z} \times \mathbf{E}, f \rangle$ from $\mathbf{R}_0$, it is assumed that:

(1) $\boldsymbol{A}$ is fully observable and fully controllable;
(2) the set of criterial values $\mathbf{E}$ is $\{0, 1\}$;
(3) for any internal state of $\boldsymbol{A}$ at a step i, an optimal control action $U_i = U_{\text{opt}} \in \mathbf{U}$ exists such that the criterial value $e_{i+1} = 0$.

It is clear from the definition of an optimal control function (see section A1) that it should be a TRF. A question arises whether the set of all TRFs is sufficiently rich for providing an optimal control function for every (constructive) control object. The following lemma gives a positive answer to it.

LEMMA 1. *For any controlled object from* $\mathbf{R}_0$, *an optimal control function exists in the form of a TRF.*

Proof. Let a controlled object $\boldsymbol{A} = \langle \mathbf{U}, \mathbf{S}, \mathbf{X} = \mathbf{Z} \times \mathbf{E}, f \rangle$ be given. At an arbitrary step i, it produces an output X_i. Since the object is fully observable by definition, a TRF h exists such that $S_i = h(X_i)$. Then, due to condition (2), an optimal control action can be computed in the following way: $U_{\text{opt}} = \mu U[f^{\text{E}}(U, h(X_i)) = 0]$, where μ is the operator of minimalisation and f^{E} describes the criterial value output. This optimal control law is a TRF according to the condition (3) in the above definition of the class $\mathbf{R}_0$ of controlled objects. □

The next lemma asserts that any TRF can be presented as the *only* optimal control function for a certain controlled object.

LEMMA 2. *For any given TRF g that satisfies condition* (2), *there exists a controlled object in RO, such that, regardless of its initial state, it can be controlled in an optimal way by and only by g.*

Proof. Let a total computable function $g : \mathbf{X} \to \mathbf{U}$ be given, where $\mathbf{X} = \mathrm{N}^m$. A construction of a desired controlled object is based on the two following ideas. First, at each step i, the criterial variable e should be equal to zero if and only if $U_i = g(X_i)$. Second, the controlled object's functioning should be organized in such a way that, regardless of its current internal state, any point from $\mathbf{X}$ appears at least once on the object's output. This would ensure that, whenever a control system $\boldsymbol{C} = \langle \mathbf{X}, \oslash, \mathbf{U}, r \rangle$ starts working, if there exists a point $X_j \in \mathbf{X}$ such that $r(X_j) \neq g(X_i)$ and therefore $e_j = 1$, it will be encountered in a finite number of steps and thus the non-optimality of r will be revealed.

Let $\boldsymbol{A} = \langle \mathbf{U}, \mathbf{S}, \mathbf{Y} = \mathbf{X} \times \mathbf{E}, f \rangle$ be a controlled object. The first requirement is easily satisfied by setting

$$e_{i+1} = f^{\mathrm{E}}(U_i, S_i) = \begin{cases} 0, & \text{if } U_i = g(X_i), \\ 1, & \text{if } U_i \neq g(X_i). \end{cases}$$

Note that this definition is very similar to the analogous part of Gold's paradigm for language learning (Gold, 1967). The second requirement can be satisfied in several different ways. One of them is the following. Let $\mathbf{S}$ be equal to N, and let $f^{\mathrm{S}}(U, s)$ be defined as $s + 1$, i.e. $s_{i+1} = s_i, +1$. A definition for $f^{\mathrm{X}}(U, s)$ has to be somewhat more complex. Let the k-th element of $X_{i+1} = (x_1, x_2, \ldots, x_m)$ be obtained from s_i by finding the power of the k-th prime number in the representation of s_i as a product of prime numbers. Since such representation is unique for any natural number, we have a functional relationship between s and Z, which defines $f^{\mathrm{X}}(U, s)$. For any $X = (x_1, x_2, \ldots, x_m)$, there is an infinite number of natural numbers that can be presented as $P_1^{x_1} \cdot P_2^{x_2} \cdot \ldots \cdot p_m^{x_m} \cdot q$ where p_j is they jth prime number, and q is a number that is not a multiple of any one of the prime numbers $p_1, p_2, \ldots, p_m$. Thus, the second requirement is satisfied, and a construction of a desired controlled object is finished, which proves the Lemma. □

THEOREM 1. *A universal learning function* (*ULF*) *for* $\mathbf{R}_0$ *is not computable.*

Proof. Let us assume the opposite. Then a universal learning algorithm (ULA) exists for computing the ULF. It will be shown that such an algorithm can be used for the enumeration of the set of all TRFs. Suppose that we can enumerate all controlled objects from the class R_0. Then we can perform that enumeration while connecting the objects to a ULS and taking function

codes from the output of its learning subsystem. According to Lemma 1, any such object can be optimally controlled by a TRF, and, according to Lemma 2, for any given TRF f there exists an object such that it can be controlled in an optimal way by f only. An existence of a ULA means that the corresponding ULS will produce a code of f (by computing the learning function, see ULS definition) in a finite number of steps after having been connected to the object. Thus a desired enumeration of all TRFs can be obtained.

However, the enumeration of controlled objects is not less problematic than the enumeration of TRFs. Fortunately, it appears that we actually do not need to enumerate the objects themselves. A ULA computes the learning function $L : \mathbf{X}^* \rightarrow \mathrm{N}$ translating finite sequences of a current controlled object's outputs into the code of an optimal control TRF. In other words, for any given TRF f, there is a finite sequence of $\mathbf{X}$ elements such that if arrived at the ULS's input, would be converted by the ULA into the code of f. Since $\mathbf{X}^*$ is effectively enumerable, we can enumerate this set instead of the class $\mathbf{R}_0$, thus obtaining an algorithm enumerating all TRFs as a combination of an algorithm enumerating $\mathbf{X}^*$ with the ULA.

It is well known in the theory of recursive functions, that the set of all TRFs is not recursively enumerable. Thus, the theorem is proved by contradiction. □

Conclusion: a ULS in general is *not* constructive.

A4. SUFFICIENCY OF TRF ENUMERATION

It was proved above that enumeration of the set of all TRFs is necessary for ULS functioning. Is it also sufficient? A positive answer to this is given by the following theorem.

THEOREM 2. *It is possible to create a ULS based on a mechanism that enumerates the set of all TRFs in* $\mathbf{R}_0$.

Proof. Let us assume the existence of such a mechanism. Then it can be used in the following way. After the code of a TRF (representing the control function) is generated, a universal algorithm starts computing this function. If the criterial variable becomes non-zero at a certain step i, the generation of the next TRF is initiated. Otherwise, the current TRF is kept for processing the next input from the controlled object. Lemma 1 guarantees that an optimal control function can be found among TRFs for any controlled object. Thus, if an optimal control function exists, it will be found after a finite number of steps, which proves the theorem. □

This positive result is proved here for the particular case of the defined above narrow class $\mathbf{R}_0$ of fully observable and fully controllable objects. One can show without much difficulty that Theorem 2 also holds in the most general case (i.e. for the entire class **R**). In order to demonstrate that, a notion of an *internal model* of the controlled object in a learning control system should be introduced as a subsystem of the latter. It can be proved that such a subsystem can be effectively created based on the assumption of Theorem 2. Then a ULS can be constructed using the operator of perpetual search for a global extremum of a total function and the assumed mechanism of enumerating TRFs. A complete proof, however, is too lengthy to be included in this paper.

It is important to note that, although the finiteness of search for an optimal control function is guaranteed by Theorem 2, there is no criterion for determining the exact moment at which an optimal algorithm is found. Stated differently, there is no basis for stopping learning, because there is no guarantee that a criterial signal will not appear after processing the next input from the controlled object, no matter how long the control system has been working already.

A5. THE AXIOM OF CHOICE AND UNIVERSAL LEARNING

The 'classical' mathematics (including integral and differential calculi and all other tools the functional analysis is based on) is separated from the constructive mathematics mainly by the axiom of choice. That axiom asserts a possibility to choose an element from a set whose description (not necessarily constructive) is given. Despite the superficial simplicity of this statement, it is in fact a very strong assumption. It almost immediately follows from the axiom of choice that any given set can be arranged in a linear order, which means that any countable set can be enumerated. Let us assume that we have already chosen several elements (set A_m) from a given set A. The axiom says that we can choose an element from $A \backslash A_m$ (A_m complement to A), i.e. a new element from A. By continuing this process, we can actually enumerate elements from A. To prove the existence of a ULS, we need only the axiom of choice reduced to countable sets:

THEOREM 3. *A Universal Learning System exists in the realm of classical mathematics.*

Proof. The set of all PRF's is countable, since it is effectively enumerable. Any set of TRF's (including the set of all TRF's) is also countable, as a subset of the above set. Consequently, it is possible to enumerate any set of TRFs, according to the axiom of choice for countable sets. Then a ULS for this set exists according to the Theorem 2. □

References

Amari, S. (1991), 'Mathematical Theory of Neural Learning', *New Generation Computing* 8, pp. 281–294.

Baev, K.V. and Shimansky, Y.P. (1992), 'Principles of Organization of Neural Systems Controlling Automatic Movements in Animals', *Progress in Neurobiology* 39, pp. 45–112 .

Barto, A.G., Sutton, R.S. and Anderson, C.W. (1983), 'Neuronlike Adaptive Elements that Can Solve Difficult Learning Control Problems', *IEEE Transactions on Systems, Man, and Cybernetics*, SMX-13, pp. 834–846.

Bates, E.A. and Elman, J.L. (1996), 'Learning Rediscovered', *Science* 274, pp. 1849–1850.

Bell, J.S. (1966), 'On the Problem of Hidden Variables in Quantum Mechanics', *Reviews of Modern Physics* 38, pp. 447–452.

Bertsekas, D.P. (2000), *Dynamic Programming and Optimal Control*, Belmont: Athena Scientific.

Bertsekas, D.P. and Tsitsiklis, J.N. (1996), *Neuro-Dynamic Programming*, Belmont: Athena Scientific.

Bhushan, N. and Shadmehr, R. (1999), 'Computational Nature of Human Adaptive Control During Learning of Reaching Movements in Force Fields', *Biological Cybernetics* 81, pp. 39–60.

Calvin, W.H. and Bickerton, D. (2000), *Lingua ex machina: Reconciling Darwin and Chomsky with the Human Brain*, Cambridge, MA: MIT Press.

Chomsky, N. (1966), *Topics in the Theory of Generative Grammar*, The Hague: Mouton.

Chomsky, N. (1980), 'On Cognitive Structures and Their Development: A Reply to Piaget', in M. Piatelli-Palmarini, ed., *Language and Learning*, Cambridge, MA: Harvard University Press, pp. 35–52 .

Cutland, N. (1980), *Computability*, Cambridge: Cambridge University Press.

Cziko, G. (1995), *Without Miracles. Universal Selection Theory and the Second Darwinian Revolution*, Cambridge, MA: MIT Press.

Elman, J.L., Bates, E.A., Hohnson, M.H., Karmiloff-Smith, A., Parisi, D. and Plunkett, K. (1996), *Rethinking Innateness*, Cambridge, MA: MIT Press.

Fodor, J. (1980), 'Fixation of Belief and Concept Acquisition', in M. Piatelli-Palmarini, ed., *Language Learning* Cambridge, MA: Harvard University Press, pp. 143–149.

Gerdes, V.G. and Happee, R. (1994), 'The Use of Internal Representation in Fast Goal-Directed Movements: A Modeling Approach', *Biological Cybernetics* 70, pp. 513–524.

Gold, E. (1967), 'Language Identification in the Limit', *Information and Control* 10, pp. 447–474.

Hebb, D.O. (1949), *The Organization of Behavior*, New York: Wiley.

Hutter, M. (2001), 'Towards a Universal Theory of Artificial Intelligence Based on Algorithmic Probability and Sequential Decisions', in D.L. Raedt and P. Flash, eds., *Proceedings of the 12th European Conference of Machine Learning*, Berlin: Springer-Verlag, pp. 226–238.

Johansson, R. and Magnusson, M. (1991), 'Optimal Coordination and Control of Posture and Locomotion', *Mathematical Biosciences* 103, pp. 203–244 .

Kieu, T.D. (2003), 'Quantum Algorithm for Hilbert's Tenth Problem', *International Journal Theoretical Physics* 42, pp. 1461–1478.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983), 'Optimization by Simulated Annealing', *Science* 228, pp. 671–680.

Lan, N. (1997), 'Analysis of an Optimal Control Model of Multi-Joint Arm Movements', *Biological Cybernetics* 76, pp. 107–117.

Langley, P. (1996), *Elements of Machine Learning*, San Francisco: Morgan Kaufmann.

Lawrie, I.D. (1990), *A Unified Grand Tour of Theoretical Physics*, Bristol: Institute of Physics Publishing.

Leeuw, K.D., Moore, E.F., Shannon, C.E. and Shapiro, N. (1956), 'Computability by Probabilistic Machines', in C.E. Shannon and J. McCarthy, eds., *Automata Studies*, Princeton: Princeton University Press, pp. 183–212.

Li, M. and Vitanyi, P.M.B. (1997), *An Introduction to Kolmogorov Complexity and its Applications*, Berlin: Springer.

Mayr, E. (1982), 'Teleological and Teleonomical: A New Analysis', in H.C. Plotkin, ed., *Learning, Development, and Culture*, New York: Wiley.

McCulloch, W.S. and Pitts, W.H. (1943), 'A Logical Calculus of the Idea Immanent in Nervous Activity', *Bulletin Mathematical Biophysics* 5, pp. 115– 133.

Mitchell, M. (1996), *An Introduction to Genetic Algorithms*, Cambridge, MA: MIT Press.

Penrose, R. (1994), *Shadows of the Mind*, New York: Oxford University Press.

Piaget, J. (1972), *The Principles of Genetic Epistemology*, New York: Basic Books.

Piaget, J. (1980), 'The Psychogenesis of Knowledge and its Epistemological Significance', in M. Piatelli-Palmarini, ed., *Language and Learning*, Cambridge, MA: Harvard University Press, pp. 23–34.

Piatelli-Palmarini, M. (1989), 'Evolution, Selection and Cognition: From ''Learning'' to Parameter Setting in Biology and the Study of Language', *Cognition* 31, pp. 1–44 .

Pla-Lopez, R. (1988), 'Introduction to a Learning General Theory', *Cybernetics and Systems: An International Journal* 19, pp. 411–429.

Pontryagin, L.S., Boltyanskii, V.G., Gamkrelidze, R.V. and Mishchenko, E.F. (1962), *The Mathematical Theory of Optimal Processes*, New York: Interscience.

Rosenblatt, F. (1962), *Principles of Neurodynamics*, New York: Spartan Books.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986), 'Learning Representations by Back-Propagating Errors', *Nature* 323, pp. 533–536.

Shimansky, Y.P. (2000), 'Spinal Motor Control System Incorporates an Internal Model of Limb Dynamics', *Biological Cybernetics* 83, pp. 379–389.

Shimansky, Y.P., Kang, T. and He, J. (2004), 'A Novel Model of Motor Learning Capable of Developing an Optimal Movement Trajectory On-Line from Scratch', *Biological Cybernetic*s 90, pp. 133–145.

Shoucri, R.M. (1991), 'Pump Function of the Heart as an Optimal Control Problem', *Journal of Biomedical Engineering* 13, pp. 384–390.

Solomonoff, R.J. (1964), 'A Formal Theory of Inductive Inference', *Information and Control* 7, pp. 1–2.

Sutton, R.S. and Barto, A.G. (1998), *Reinforcement Learning: An Introduction*, Cambridge, MA: MIT Press.

Uspensky, V. and Semenov, A. (1993), *Algorithms: Main Ideas and Applications*, Dordrecht: Kluwer Academic Publishers.

Wolpert, D.M., Ghahramani, Z. and Jordan, M.I. (1995), 'An Internal Model for Sensorimotor Integration', *Science* 269, pp. 1880–1882.