Entropy **2012**, *14*, 2066-2080; doi:10.3390/e14112066

OPEN ACCESS

entropy

ISSN 1099-4300

www.mdpi.com/journal/entropy

Article

Unconventional Algorithms: Complementarity of Axiomatics and Construction

Gordana Dodig Crnkovic [1,*] and Mark Burgin [2]

1 Department of Computer Science and Networks, School of Innovation, Design and Engineering Mälardalen University, Västerås, 72123, Sweden

2 Department of Mathematics, UCLA, Los Angeles, CA 90095, USA; E-Mail: mburgin@math.ucla.edu (M.B.)

* Author to whom correspondence should be addressed; E-Mail: gordana.dodig-crnkovic@mdh.se; Tel.: +46 21 15 17 25.

Received: 20 August 2012; in revised form: 23 September 2012; / Accepted: 19 October 2012 / Published: 25 October 2012

Abstract: In this paper, we analyze axiomatic and constructive issues of unconventional computations from a methodological and philosophical point of view. We explain how the new models of algorithms and unconventional computations change the algorithmic universe, making it open and allowing increased flexibility and expressive power that augment creativity. At the same time, the greater power of new types of algorithms also results in the greater complexity of the algorithmic universe, transforming it into the algorithmic multiverse and demanding new tools for its study. That is why we analyze new powerful tools brought forth by local mathematics, local logics, logical varieties and the axiomatic theory of algorithms, automata and computation. We demonstrate how these new tools allow efficient navigation in the algorithmic multiverse. Further work includes study of natural computation by unconventional algorithms and constructive approaches.

Keywords: unconventional computing; computation beyond the Turing limit; axiomatic *vs.* constructive models; unconventional models of computation

---

1. Introduction

The development of computer science and information technology brought forth a diversity of novel algorithms and algorithmic schemas, unconventional computations and nature-inspired

processes, advanced functionality and conceptualizations. The field of information processing and information sciences encountered the stage of transition from classical subrecursive and recursive algorithms, like finite automata, recursive functions or Turing machines, to new super-recursive algorithms, such as inductive Turing machines or limiting recursive functions.

Super-recursive algorithms controlling and directing unconventional computations exceed the boundary set by Turing machines and other recursive algorithms, resulting in an open algorithmic universe and revealing new levels of creativity. As the growth of possibilities involves a much higher complexity of the new, open world of super-recursive algorithms, unconventional computations, innovative hardware and advanced organization, we discuss means of navigation in this open algorithmic world.

The paper is organized as follows: in Section 2, we compare characteristics of local and global mathematics, explaining why local mathematics allows better modelling of reality. Section 3 addresses relationships between local mathematics, local logics and logical varieties, while Section 4 offers the discussion of projective mathematics versus reverse mathematics versus classical mathematics. Section 5 answers the question of how to navigate in the algorithmic multiverse. Finally, Section 6 presents our conclusions and provides directions for future work.

2. Local Mathematics *vs.* Global Mathematics

As an advanced knowledge system, mathematics exists as an aggregate of various mathematical fields. If at the beginning there were only two fields—arithmetics and geometry, now there are hundreds of mathematical fields and subfields. However, mathematicians always believed in mathematics as a unified system striving to build common and in some senses absolute foundations for all mathematical fields and subfields. At the end of the 19th century, mathematicians came very close to achieving this goal as the emerging set theory allowed the construction of all mathematical structures using only sets and operations with sets. However, in the 20th century, it was discovered that there are different set theories. This brought some confusion and attempts to find the "true" set theory.

To overcome this confusion, Bell [1] introduced in 1986 the concept of local mathematics. The fundamental idea was to abandon the unique, absolute universe of sets central to the orthodox set-theoretic account of the foundations of mathematics, replacing it by a plurality of local mathematical frameworks. Bell suggested taking elementary topoi as such frameworks, which would serve as local replacements for the classical universe of sets. Having sufficient means for developing logic and mathematics, elementary topoi possess a sufficiently rich internal structure to enable a variety of mathematical concepts and assertions to be interpreted and manipulated. Mathematics interpreted in any such framework is called *local mathematics* and admissible transformation between frameworks amounts to a (definable) *change of local mathematics.* With the abandonment of the absolute universe of sets, mathematical concepts in general lose absolute meaning, while mathematical assertions liberate themselves from absolute truth values. Instead they possess such meanings or truth values only *locally*, *i.e.*, *relative* to local frameworks. This means that the *reference* of any mathematical concept is accordingly not fixed, but *changes* with the choice of local mathematics.

It is possible to extend the approach of Bell in three directions. First, we can use an arbitrary category as a framework for developing mathematics. When an internal structure of such a framework

is meager, the corresponding mathematics will also be indigent. Second, it is possible to take a theory of some structures instead of the classical universe of sets and develop mathematics within that framework without reference to the universal framework. Third, as we know, there are different axiomatizations of set theory. Developed axiomatics are often incompatible, e.g., axiomatics in which the Continuum Hypothesis is true and axiomatics where it is false. Thus, developing mathematics based on one such axiomatics also results in a local mathematics.

A similar situation emerged in computer science, where mathematics plays a pivotal role. Usually, to study properties of computers, computer networks and computational processes and elaborate more efficient applications, mathematicians and computer scientists use mathematical models. There is a variety of such models: Turing machines of different kinds (with one tape and one head, with several tapes, with several heads, with n-dimensional tapes, non-deterministic, probabilistic, and alternating Turing machines, Turing machines that take advice and Turing machines with oracle, *etc.* [2]), Post productions, partial recursive functions, neural networks, finite automata of different kinds (automata without memory, autonomous automata, accepting automata, probabilistic automata, *etc.*), Minsky machines [3], normal Markov algorithms [4], Kolmogorov algorithms [5], formal grammars of different kinds (regular, context free, context sensitive, phrase-structure, *etc.*), Storage Modification Machines or simply Shönhage machines [6], Random Access Machines (RAM) [7], Petri nets [8], which like Turing machines have several forms (ordinary, regular, free, colored, self-modifying, *etc.*), and so on. All these models are constructive, *i.e.*, they have tractable explicit descriptions and simple rules for operation. Thus, the constructive approach dominates in computer science.

This diversity of models is natural and useful because each type is suited to a particular type of problem. In other words, the diversity of problems that are solved by computers gives rise to a corresponding diversity of models. For example, general problems of computability involve such models as Turing machines and partial recursive functions. Finite automata are used for text search, lexical analysis, and construction of semantics for programming languages. In addition, different computing devices demand corresponding mathematical models. For example, universal Turing machines and inductive Turing machines allow one to investigate characteristics of conventional computers [2]. Petri nets are useful for modeling and analysis of computer networks, distributed computation, and communication processes [9]. Finite automata model computer arithmetic. Neural networks reflect properties of the brain. Abstract vector and array machines model vector and array computers [2].

To utilize some models that are related to a specific type of problem, we need to know their properties. In many cases, different classes of models have the same or similar properties. As a rule, such properties are proved for each class separately. Thus, alike proofs are repeated many times in similar situations involving various models and classes of algorithms.

In contrast to this, the *projective* (also called *multiglobal*) *axiomatic theory* of algorithms, automata and computation suggests a different approach [10]. Assuming some simple basic conditions (in the form of postulates, axioms and conditions), many profound and far-reaching properties of algorithms are derived in this theory. This allows one, when dealing with a specific model, not to prove this property, but only to check the conditions from the assumption, which is much easier than to prove the property under consideration. In such a way, we can derive various characteristics of types of computers and software systems from the initial postulates, axioms and conditions.

The projective approach in computer science has its counterpart in mathematics, where systems of unifying properties have been used for building new encompassing structures, proving indispensable properties in these new structures and projecting these properties on the encompassed domains. Such projectivity has been explicitly utilized in category theory, which was developed and utilized with the goal of unification [11].

Breaking the barrier of the Church-Turing Thesis drastically increased the variety of algorithmic model classes and changed the algorithmic universe of recursive algorithms to the multiverse of super-recursive algorithms [2], which consists of a plurality of local algorithmic universes. Each class of algorithmic model forms a local algorithmic universe, providing means for the development of local computer science in general and a local theory of algorithms in particular.

Local mathematics brings forth local logics because each local mathematical framework has its own logic and it is possible that different frameworks have different local logics.

3. Logical Varieties as a Unification of Local Logics

Barwise and Seligman [12] developed a theory of information flow reflecting the dynamics of information processing systems. In this theory, the concept of a *local logic* plays a fundamental role in the modeling of commonsense reasoning, which is an important kind of information processing.

The basic concept of the information flow theory is a *classification*. A natural interpretation of a classification, which is a typical named set [13], is a representation of some domain in the physical or abstract world by a system of symbols, which denote types of objects from the represented domain. Each local logic corresponds to a definite classification describing properties of the domain and the classification in a logical language and allowing one to deduce previously unknown properties. This implies a natural condition that each domain has its own local logic and different domains may have different local logics.

In a similar way, each class of algorithms from the algorithmic multiverse, as well as a constellation of such classes, forms a local algorithmic universe, which has a corresponding local logic. These logics may be essentially different. For instance, taking two local algorithmic universes formed by such classes as the class ***T*** of all Turing machines and the class ***TT*** of all total, *i.e.*, everywhere defined, Turing machines, we find that the first class satisfies the axiom of universality [10], which affirms existence of a universal algorithm, *i.e.*, a universal Turing machine in this class. However, the class ***TT*** does not satisfy this axiom [10].

Barwise and Seligman [12] assumed that the totality of local logics forms a set. However, analyzing the system of local logics, it is possible to see that there are different relations between them and it would be useful to combine these logics in a common structure. As is explained in [13], local logics form a deductive logical variety or a deductive logical prevariety, which were introduced and studied in [14] as a tool to work with inconsistent systems of knowledge. Logical varieties and prevarieties provide a unified system of logical structures, in which local logics are naturally integrated.

Minsky [15] was one of the first AI researchers who brought attention to the problem of inconsistent knowledge. He wrote that consistency is a delicate concept that assumes the absence of contradictions in systems of axioms. Minsky also suggested that in artificial intelligence (AI) systems this assumption was superfluous because there were no completely consistent AI systems. In his

opinion, it is important to understand how people solve paradoxes, find a way out of a critical situation, and learn from their own or others' mistakes or how they recognize and exclude different inconsistencies. In addition, Minsky [16] suggested that consistency and effectiveness may well be incompatible. He further writes [17]: "An entire generation of logical philosophers has thus wrongly tried to force their theories of mind to fit the rigid frames of formal logic. In doing that, they cut themselves off from the powerful new discoveries of computer science. Yes, it is true that we can describe the operation of a computer's hardware in terms of simple logical expressions. But no, we cannot use the same expressions to describe the meanings of that computer's output -- because that would require us to formalize those descriptions inside the same logical system. And this, I claim, is something we cannot do without violating that assumption of consistency." Minsky [17] continues, "In summary, there is no basis for assuming that humans are consistent - nor is there any basic obstacle to making machines use inconsistent forms of reasoning". Moreover, it has been discovered that not only human knowledge but also representations/models of human knowledge (e.g., large knowledge bases) are inherently inconsistent [18]. Logical varieties or prevarieties provide powerful tools for working with inconsistent knowledge.

There are different types and kinds of logical varieties and prevarieties: *deductive* or *syntactic varieties* and *prevarieties*, *functional* or *semantic varieties* and *prevarieties* and *model* or *pragmatic varieties* and *prevarieties*. Syntactic varieties, prevarieties, and quasi-varieties (which were introduced in [19]) are built from logical calculi as building blocks. Semantic varieties and prevarieties (which were introduced and studied in [20]) are built from logics, while model varieties and prevarieties (also introduced and studied in [20]) are built from separate logical models.

Let us consider a logical language L, an inference language R, a class **K** of syntactic logical calculi, a set Q of inference rules ($Q \subseteq R$), and a class **F** of partial mappings from L to L.

A triad **M** = (A, H, M), where A and M are sets of expressions that belong to L (A consists of axioms and M consists of theorems) and H is a set of inference rules, which belong to the set R, is called:

(1) a *projective syntactic* (**K,F**)-*prevariety* if there exists a set of logical calculi $\boldsymbol{C}_i = (A_i\ , H_i\ , T_i\ )$ from **K** and a system of mappings f_i: $A_i \to L$ and g_i: $M_i \to L$ ($i \in I$) from **F** in which A_i consists of all axioms and M_i consists of all theorems of the logical calculus $\boldsymbol{C}_i$, and for which the equalities $A = \cup_{i\in I} f_i(A_i)$, $H = \cup_{i\in I} H_i$ and $M = \cup_{i\in I} g_i(M_i)$ are valid (it is possible that $\boldsymbol{C}_i = \boldsymbol{C}_j$ for some $i \neq j$).

(2) a *projective syntactic* (**K,F**)-*variety* with the depth k if it is a projective syntactic (**K,F**)-quasi-prevariety and for any i_1 , i_2 , i_3 , … , $i_k \in I$ either the intersections $\cap_{j=1}^{k} f_{ij}(A_{ij})$ and $\cap_{j=1}^{k} g_{ij}(T_{ij})$ are empty or there exists a calculus $\boldsymbol{C} = (A, H, T)$ from **K** and projections f: $A \to \cap_{j=1}^{k} f_{ij}(A_{ij})$ and g: $N \to \cap_{j=1}^{k} g_{ij}(M_{ij})$ from **F** where $N \subseteq T$;

(3) a *syntactic* **K**-prevariety if it is a projective syntactic (**K,F**)-prevariety in which $M_i = T_i$ for all $i \in I$ and all mappings f_i and g_i that define **M** are bijections on the sets A_i and M_i , correspondingly;

(4) a *syntactic* **K**-*variety* if it is a projective syntactic (**K,F**)- variety in which $M_i = T_i$ for all $i \in I$ and all mappings f_i and g_i that define **M** are bijections on the sets A_i and M_i , correspondingly.

The calculi $\boldsymbol{C}_i$ used in the formation of the prevariety (variety) **M** are called *components* of **M**.

Note that different components of deductive logical varieties and prevarieties can not only contain distinct axioms and theorems but also employ distinctive deduction rules. For instance, one component can use classical deduction, while another component of the same variety can be based on a relevant logic.

We see that the collection of mappings f_i and g_i makes a unified system called a prevariety or quasi-prevariety out of separate logical calculi $\boldsymbol{C}_i$, while the collection of the intersections $\bigcap_{j=1}^{k} f_{ij}(A_{ij})$ and $\bigcap_{j=1}^{k} g_{ij}(T_{ij})$ makes a unified system called a variety out of separate logical calculi $\boldsymbol{C}_i$. For instance, mappings f_i and g_i allow one to establish a correspondence between norms/laws that were used in one country during different periods of time or between norms/laws used in different countries. In a similar way, relations between components of logical varieties and prevarieties allow one to establish a correspondence between properties of different models of computation and algorithmic classes.

The main goal of syntactic logical varieties is in presenting sets of formulas as a structured logical system using logical calculi, which have means for inference and other logical operations. Semantically, it allows one to describe a domain of interest, e.g., a database, knowledge of an individual or the text of a novel, by a syntactic logical variety dividing the domain in parts that allow representation by calculi.

In comparison with varieties and prevarieties, logical quasi-varieties and quasi-prevarieties studied in [20] are not necessarily closed under logical inference. This trait allows better flexibility in knowledge representation.

While syntactic logical varieties and prevarietis synthesize local logics in a unified system, semantic logical varieties and prevarieties studied in [20] unify local mathematics forming a holistic realm of mathematical knowledge. Local meaning of mathematical concepts is defined by model logical varieties and prevarieties and is relative with respect to each component of the corresponding variety or prevariety. In the context of local mathematics, mathematical assertions liberate themselves from absolute truth values acquiring relative truth values specifically defined by each component of the corresponding semantic logical variety or prevariety.

In addition, syntactic logical varieties and prevarieties found diverse applications to databases and network technology providing tools for working with inconsistency, imprecision, vagueness, non-monotonic inference, knowledge base unification and database integration (cf., for example, [21]).

4. Projective Mathematics *vs.* Reverse Mathematics *vs.* Classical Mathematics

According to Suppe [22, p. 9], "axiomatization is a formal method for specifying the content of a theory wherein a set of axioms is given from which the remaining content of the theory can be derived deductively as theorems. The theory is identified with the set of axioms and its deductive consequences, which is known as the closure of the axiom set. The logic used to deduce theorems may be informal, as in the typical axiomatic presentation of Euclidean geometry; semiformal, as in reference to set theory or specified branches of mathematics; or formal, as when the axiomatization consists in augmenting the logical axioms for first-order predicate calculus by the proper axioms of the theory."

Mathematics suggests an approach for knowledge unification, namely, it is necessary to find axioms that characterize all theories in a specific area and to develop the theory in an axiomatic context. This approach has worked extremely well in a variety of mathematical fields, providing rigorous tools for mathematical exploration.

Axiomatization has often been used in physics (Hilbert's sixth problem refers to axiomatization of branches of physics in which mathematics is prevalent and researchers found that, for example, finding

the proper axioms for quantum field theory is still an open and difficult problem in mathematics), biology (according to Britannica, [23] the most enthusiastic proponent of this approach, the British biologist and logician Joseph Woodger, attempted to formalize the principles of biology—to derive them by deduction from a limited number of basic axioms and primitive terms—using the logical apparatus of the *Principia Mathematica* by Whitehead and Bertrand Russell), and some other areas, such as philosophy or technology. It is interesting that the axiomatic approach was also used in areas that are very far from mathematics. For instance, Spinoza used this approach in philosophy, developing his ethical theories and writing his book *Ethics* in the axiomatic form. More recently, Kunii [24] developed an axiomatic system for cyberworlds.

Since the advent of computers, deductive reasoning and axiomatic exposition have been delegated to computers, which performed theorem-proving, while the axiomatic approach has come to software technology and computer science. Logical tools and axiomatic description have been used in computer science for different purposes. For instance, Manna [25] built an axiomatic theory of programs, while Milner [26] developed an axiomatic theory of communicating processes. An axiomatic description of programming languages was constructed by Meyer and Halpern [27]. Many researchers have developed different kinds of axiomatic recursion theories (cf., for example [28–33]).

However, in classical mathematics, axiomatization has a global character. Mathematicians tried to build a unique axiomatics for the foundations of mathematics. Logicians working in the theory of algorithms tried to find axioms comprising all models of algorithms.

This is the classical approach – axiomatizing the studied domain and then deducing theorems from axioms. All classical mathematics is based on deduction as a method of logical reasoning and inference. *Deduction* is a type of reasoning process that constructs and/or evaluates *deductive arguments*, where the conclusion follows from the premises with logical necessity. In logic, an argument is called deductive when the truth of the conclusion is purported to follow necessarily or be a logical consequence of the assumptions. Deductive arguments are said to be valid or invalid, but never true or false. A deductive argument is valid if and only if the truth of the conclusion actually does follow necessarily from the assumptions. A valid deductive argument with true assumptions is called sound. A deductive argument which is invalid or has one or more false assumptions or both is called unsound. Thus, we may call classical mathematics by the name *deductive mathematics*.

The goal of deductive mathematics is to deduce theorems from axioms. Deduction of a theorem is also called proving the theorem. When mathematicians cannot prove some interesting and/or important conjecture, researchers with a conventional thinking try to prove that the problem is unsolvable in the existing framework. Creative explorers instead invent new structures and methods, construct new framework, introducing new axioms to solve the problem.

Some consider deductive mathematics as a part of axiomatic mathematics, assuming that deduction (in a strict sense) is possible only in an axiomatic system. Others treat axiomatic mathematics as a part of deductive mathematics, assuming that there are other inference rules besides deduction.

While deductive mathematics is present in and actually dominates all fields of contemporary mathematics, reverse mathematics is the branch of mathematical logic that seeks to determine what are the minimal axioms (formalized conditions) needed to prove a particular theorem [34,35]. This direction in mathematical logic was founded by [28,36]. The method can briefly be described as going backwards from theorems to the axioms necessary to prove these theorems in some logical system [37]. It

turns out that over a weak base theory, many mathematical statements are equivalent to the particular new postulate needed to prove them. This methodology contrasts with the ordinary mathematical practice where theorems are deduced from *a priori* assumed axioms.

Reverse mathematics was prefigured by some results in set theory, such as the classical theorem that states that the axiom of choice, well-ordering principle of Zermelo, maximal chain priciple of Hausdorff, Zorn's lemma [38], and statements of the vector basis theorem [39] and Tychonov product theorem [40] are equivalent over ZF set theory (Howard and Rubin, 1998) [41]. The goal of reverse mathematics, however, is to study ordinary theorems of mathematics rather than possible axioms for set theory. A sufficiently weak base theory is adopted (usually, it is a subsystem of second-order arithmetic) and the search is for minimal additional axioms for deducing some interesting/important mathematical statements. It has been found that in many cases these minimal additional axioms are equivalent to the particular statements they are used to prove.

Projective mathematics is a branch of mathematics similar to reverse mathematics, which aims to determine what simple conditions are needed to prove the particular theorem or to develop a particular theory. However, there are essential differences between these two directions: reverse mathematics is aimed at a logical analysis of mathematical statements, while projective mathematics is directed at making the scope of theoretical statements in general and mathematical statements in particular much larger whilst extending their applications. As a result, instead of proving similar results in various situations, it becomes possible to prove a corresponding general result in the axiomatic setting and to ascertain validity of this result for a particular case by demonstrating that all axioms (conditions) used in the proof are true for this case. In this way the general result is projected on different situations. This direction in mathematics was founded by Burgin [10]. This approach contrasts with conventional (deductive) mathematics where axioms describe some area or type of mathematical structures, while theorems are deduced from a priori assumed axioms.

Projective mathematics has its precursor in such results as the extension of many theorems initially proved for numerical functions to functions in metric spaces [42], or generalizations of properties of number systems to properties of groups, rings and other algebraic structures [39].

Here we describe how projective mathematics is used for exploration of computations controlled by algorithms and realized by automata. In this application of projective mathematics, the goal is to find some simple properties of computations, algorithms and automata in general, to present these properties in the form of axioms, and to deduce from these axioms theorems that describe much more profound and sophisticated properties of computations, algorithms and automata. This allows one, taking some class **A** of algorithms, not to prove these theorems but only to check if the initial axioms are valid in **A**. If this is the case, then it becomes possible to conclude that all corresponding theorems are true for the class **A**. As we know, computer scientists and mathematicians study and utilize a huge variety of different classes and types of algorithms, automata, and abstract machines. Consequently, such an axiomatic approach allows them to obtain many properties of studied algorithms and automata in a simple and easy way.

It is possible to explain goals of classical (deductive) mathematics, reverse mathematics and projective mathematics by means of relations between axioms and theorems.

A set A of axioms can be:

(1) *Consistent* with some result (theorem) T, *i.e.*, when the theorem T is added as a new axiom, the new system remains consistent, allowing one to, in some cases, deduce (prove) this theorem.

(2) *Sufficient* for some result (theorem) T, *i.e.*, it is possible to deduce (prove) the theorem T using axioms from A.

(3) *Irreducible* with respect to some result (theorem) T, *i.e.*, the system A is a minimal set of the axiom that allows one to deduce (prove) the theorem T.

After the discovery of non-Euclidean geometries, the creation of modern algebra and the construction of set theory, classical mathematics' main interest has been to find whether a statement T has been consistent with a given axiomatic system A (the logical goal) and then to prove this statement in the context of A. Thus, classical mathematics is concerned with the first relation. Reverse mathematics, as we can see, deals with the third relation.

In contrast to this, projective mathematics is oriented at the second relation. The goal is to find some simple properties of algorithms or automata in general, to present these properties in the form of a system U of axioms, and from these axioms, to deduce theorems that describe much more profound properties of algorithms and automata. This allows one, taking some class **A** of algorithms or automata, not to prove these theorems but only to check if all axioms from the system U are valid in **A**. If this is the case, then it is possible to conclude that all corresponding theorems are true for the class **A**. As we know, computer scientists and mathematicians study and utilize a huge variety of different classes and types of algorithms, automata, and abstract machines. In such a way, the axiom system U provides a definite perspective on different classes and types of algorithms, automata, and abstract machines.

It is interesting that Bernays had a similar intuition with respect to axioms in mathematics, regarding them not as a system of statements about a subject matter but as a system of conditions for what might be called a relational structure. He wrote in [43]: "A main feature of Hilbert's axiomatization of geometry is that the axiomatic method is presented and practiced in the spirit of the abstract conception of mathematics that arose at the end of the nineteenth century and which has generally been adopted in modern mathematics. It consists in abstracting from the intuitive meaning of the terms... and in understanding the assertions (theorems) of the axiomatized theory in a hypothetical sense, that is, as holding true for any interpretation... for which the axioms are satisfied. Thus, an axiom system is regarded not as a system of statements about a subject matter but as a system of conditions for what might be called a relational structure... [On] this conception of axiomatics, ... logical reasoning on the basis of the axioms is used not merely as a means of assisting intuition in the study of spatial figures; rather, logical dependencies are considered for their own sake, and it is insisted that in reasoning we should rely only on those properties of a figure that either are explicitly assumed or follow logically from the assumptions and axioms."

It is possible to formalize the approach of projective mathematics using logical varieties. Indeed, let us take a collection C of postulates, axioms and conditions, which are formalized in a logical language as axioms. This allows us to assume that we have a logical variety M that represents a given domain D in a formal mathematical setting and contains the set C. For instance, the domain D consists of a system of algorithmic models so that the logic of each model D_i is a component M_i of M. Then we

deduce a theorem T from the statements from C. Then instead of proving the theorem T for each domain D_i, we check whether $C \subseteq M_i$. When this is true, we conclude that the theorem T belongs to the component M_i because M_i is a calculus and thus, the theorem T is valid for the model D_i. Because C usually consists of relatively simple statements, to check the inclusion $C \subseteq M_i$ is simpler than to prove T in M_i. In addition, this approach provides unification for the whole theory of algorithms, automata and computation as it explicates similarities and common traits in different algorithmic models and abstract automata.

5. How To Navigate in the Algorithmic Multiverse

It is possible to see that for a conformist, it is much easier to live in the closed algorithmic universe because all possible and impossible actions, as well as all solvable and insolvable problems can be measured against one of the most powerful and universal classes of algorithms in the algorithmic universe. This has usually been done utilizing Turing machines.

The open world provides many more opportunities for actions and problem solving, but at the same time it demands more work, more effort and even more imagination for solving problems which are insolvable in the closed algorithmic universe. Even the closed algorithmic universe contains many classes and types of algorithms, which have been studied with reference to a universal class of recursive algorithms. In some cases, partial recursive functions have been used. In other cases, unrestricted grammars have been employed. The most popular have been the utilization of Turing machines. A big diversity of new and old classes of algorithms exists that demands specific tools for exploration.

Mathematics has invented such tools and one of the most efficient for dealing with diversity is the axiomatic method. This method was also applied to the theory of algorithms, automata and computation when the axiomatic theory of algorithms, automata and computation was created [10]. In it, many profound properties of algorithms are derived based on some simple, basic conditions (in the form of postulates, axioms and conditions). Namely, instead of proving similar results in various situations, it becomes possible to prove a necessary general result in the axiomatic setting and then to ascertain the validity of this result for a particular case by demonstrating that all axioms (conditions) used in the proof are true for this case. Note that in contrast to 20th century mathematics, where projectivity was based on unifying constructions in a form of new mathematical structures [11], such as categories or heterogeneous algebras, projectivity developed in [10] in the context of computer science extracts only unifying properties without building new structures. In such a way, the general result is projected on different situations. For instance, let us consider some basic algorithmic problems inherent in computer and network functioning. One of these problems is the Fixed Output Problem. In this problem, it is necessary to find an algorithm/automaton H that for an arbitrary algorithm/automaton A from a given class **K** and arbitrary data elements b and x informs whether application of A to x gives b as the result, *i.e.*, whether $A(x) = b$.

In [4], the theorem on undecidability of the Fixed Output Problem is proved based on the projective approach. As a result, this theorem has more than 30 corollaries for various classes of algorithms (computational models), including the famous theorem about the undecidability of the halting problem for Turing machines. Another theorem on the recognizability of the Fixed Output Problem proved in [10]

has more than 20 corollaries for various classes of algorithms (computational models), such as Turing machines, random access machines, Kolmogorov algorithms, Minsky machines, partial recursive functions, inductive Turing machines of the first order, periodic evolutionary Turing machines and limiting partial recursive functions. Note that such algorithmic problems were previously studied separately for each computational model.

The axiomatic context allows a researcher to explore not only individual algorithms and separate classes of algorithms, computational models and automata but also classes of classes of algorithms, automata, and computational models and processes. As a result, the axiomatic approach goes higher in the hierarchy of computer and network models, thus reducing the complexity of their study. The suggested axiomatic methodology is applied to the evaluation of possibilities of computers, their software and their networks, with the main emphasis on such properties as computability, decidability, and acceptability. In such a way, it becomes possible to derive various characteristics of types of computers and software systems from the initial postulates, axioms and conditions.

It is also worth mentioning that the axiomatic approach allowed researchers to prove the Church-Turing Thesis for an algorithmic class that satisfies very simple initial axioms [44,45]. These axioms form a system *C* considered in the previous section and this system provides a definite perspective on different classes of algorithms, ensuring that in these classes the Church-Turing Thesis is true, *i.e.*, it is a theorem.

Moreover, the axiomatic approach is efficient in exploring features of innovative hardware and unconventional organization.

It is interesting to remark that algorithms are used in mathematics and beyond as constructive tools of cognition. Algorithms are often opposed to non-constructive, e.g., descriptive, methods used in mathematics. The axiomatic approach is essentially descriptive because axioms describe properties of the studied objects in a formalized way.

Constructive mathematics is distinguished from its traditional counterpart, axiomatic classical mathematics, by the strict interpretation of the expression "there exists" (called in logic the *existential quantifier*) as "we can construct" and show how to do this. Assertions of existence should be backed up by constructions, and the properties of mathematical objects should be decidable in a finite number of steps.

However, in some situations, descriptive methods can be more efficient and powerful than constructive tools. Language allows one to describe many more objects than it is possible to build by available tools and materials. For instance, sufficiently rich logical languages, according to the first Gödel undecidability theorem, can represent statements that are true but are not provable. That is why descriptive methods in the form of the axiomatic approach came back to the theory of algorithms, automata and computation, becoming efficient tools in computer science.

6. Conclusions and Future Work

This paper demonstrates the role of the axiomatic methods for the following paradigms of mathematics and computer science:

-*Classical mathematics,* with global axiomatization and classical logic.

-*Local mathematics,* with local axiomatization, diverse logics and logical varieties.

-*Reverse mathematics,* with axiomatic properties decomposition and backward inference.

-*Projective mathematics,* with view axiomatization, logical varieties and properties proliferation.

Here we have considered only some of the consequences of new trends in the axiomatic approach to mathematical cognition. It would be interesting to study consequences of this approach in other fields such as epistemology and computability theory. Furthermore, inasmuch as computer science is based on mathematics, the new paradigms of mathematics presented in this work form corresponding directions in computer science giving an advantage to unconventional computations and nature-inspired architectures of information processing systems. One of the novel approaches is applying the axiomatic methods of the mathematical theory on information technology [46,47].

Another important direction for future work is the study of physical systems as information processing architectures. Computations beyond the Turing model exist not only in the universe of unconventional algorithms but even in the physical universe. The idea of Pancomputationalism (Naturalist computationalism) [48,51] suggests that all of the physical universe can be modelled on different levels of organization as a network of computational processes on informational structures [48], with information defined in the sense of Informational Structural Realism, see [49].

As a consequence, unconventional computing as it appears in natural systems is developing as an important new area of constructive research. It is presented by Stepney [49] and her article in this Special Issue, Cooper [50], authors in [49–52], as well as in the work of Rozenberg and MacLennan, see [52]. Ziegler's suggestion of axiomatizing physical computational universes [53] correlates with both the natural computationalism and the approach of projective mathematics. It remains for future work to establish the connection between unconventional algorithms and natural computing.

Acknowledgments

The authors would like to thank Andree Ehresmann, Hector Zenil and Marcin J. Schroeder for useful and constructive comments on the previous version of this work and three anonymous reviewers for numerous insightful reflections and helpful advices which considerably improved the article.

References and Notes

1. Bell, J.L. From absolute to local mathematics. *Synthese* **1986**, *69*, 409–426.
2. Burgin, M. *Super-Recursive Algorithms*; Science+Business Media Inc.: New York, NY, USA, 2005.
3. Minsky, M. *Computation: Finite and Infinite Machines*; Prentice-Hall: New York/London/ Toronto, USA/UK/Canada, 1967.
4. Markov, A.A. Theory of Algorithms. In *Transactions of the Mathematical Institute of the Academy of Sciences of the USSR*; Academy of Sciences of the USSR: Moscow, USSR, 1954.
5. Kolmogorov, A.N. On the Concept of Algorithm, *Russ. Math. Surv.* **1953**, *8*, 175–176.
6. Shönhage, A. Storage Modification Machines. *SIAM. J. Comput.* **1980**, *9*, 490–508.
7. Shepherdson, J.C.; Sturgis, H.E. Computability of Recursive Functions. *J. Assoc. Comput. Mach.* **1963**, *10*, 217–255.
8. Petri, C. *Kommunikation mit Automaten*. Ph.D. Thesis, University of Bonn, Bonn, Germany, 1962.
9. Peterson, J.L. *Petri Net Theory and the Modeling of Systems*; Prentice Hall: Englewood Cliffs, New Jersey, 1981.

10. Burgin, M. *Measuring Power of Algorithms, Computer Programs, and Information Automata*; Nova Science Publishers: New York, NY, USA, 2010.
11. Ehresmann, C. Trends toward unity in mathematics. Available online: http://archive.numdam.org/ARCHIVE/CTGDC/CTGDC_1966__8_/CTGDC_1966__8__A1_0/CTGDC_1966__8__A1_0.pdf, accessed on 18 September 2012.
12. Barwise, J.; Seligman, J. Information Flow: The Logic of Distributed Systems. In *Cambridge Tracts in Theoretical Computer Science 44*; Cambridge University Press: Cambridge, UK, 1997.
13. Burgin, M. *Theory of Information: Fundamentality, Diversity and Unification*; World Scientific: Singapore, 2010.
14. Burgin, M. Knowledge in Intelligent Systems. In Proceedings of the Conference on Intelligent Management Systems, Varna, Bulgaria, 1989; pp. 281–286.
15. Minsky, M. *A Framework for Representing Knowledge*; MIT: Cambridge, MA, USA, 1974.
16. Minsky, M. Society of Mind: A Response to Four Reviews. *Artif. Intell.* **1991**, *48*, 371–396.
17. Minsky, M. Conscious Machines. In *Machinery of Consciousness*, Proceedings of the National Research Council of Canada, 75th Anniversary Symposium on Science in Society, Ottawa, Canada, 1991.
18. Delgrande, J.P.; Mylopoulos J. Knowledge Representation: Features of Knowledge. In *Fundamentals of Artificial Intelligence*; Springer: New York/Berlin/Heidelberg, USA/Germany, 1986; pp. 3–38.
19. Burgin, M.; de Vey Mestdagh, C.N.J. The Representation of Inconsistent Knowledge. In *Advanced Knowledge Based Systems, Lecture Notes in Computer Science*, *Knowlege-Based and Intelligent Information and Engineering Systems*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6882, pp. 524–537.
20. Burgin, M. Logical varieties and covarieties. In *Theoretical Problems of Mathematics and Information Sciences*; Ukrainian Academy of Information Sciences: Kiev, Ukraine, 1997; pp. 18–34. (In Russian).
21. Burgin, M. Logical Tools for Program Integration and Interoperability. In Proceedings of the IASTED International Conference on Software Engineering and Applications; MIT: Cambridge, MA, USA, 2004; pp.743–748.
22. Suppe, F. Axiomatization. In *Companion to the Philosophy of Science*, Newton-Smith, W.H., Ed.; Blackwell: Oxford, UK, 2001.
23. Biology, Philosophy of. *Encyclopædia Britannica. Encyclopædia Britannica Online Academic Edition*. Available online: http://www.britannica.com/EBchecked/topic/681551/philosophy-of-biology, accessed on 18 September 2012.
24. Kunii, T.L. The Potentials of Cyberworlds—An Axiomatic Approach. In 3rd International Conference on Cyberworlds (CW'04), Tokyo, Japan, 2004; pp. 2–7.
25. Manna, Z. *Mathematical Theory of Computation*; McGraw Hill: New York, NY, USA, 1974.
26. Milner, M. *Communication and Concurrency*; Prentice Hall: Englewood Cliffs, NJ, USA, 1989.
27. Meyer, A.R.; Halpern, J.Y. Axiomatic definitions of programming languages: A theoretical assessment (preliminary report). In Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of programming languages, Las Vegas, NV, USA, 1980; pp. 203–212.

28. Friedman, H. Axiomatic recursive function theory. In *Logic Colloquium '69*, *Studies in Logic*; North-Holland: Amsterdam, The Netherlands, 1971; pp. 113–137.
29. Gritliot, T.J. Dissecting abstract recursion. In *Generalized Recursion Theory*; North-Holland: Amsterdam, The Netherlands, 1974; pp. 405–420.
30. Fenstad, J.E. Computation theories: An axiomatic approach to recursion on general structures. In *Lecture Notes in Mathematics*; Springer: Berlin/Heidelberg, Germany, 1975; pp. 143–168.
31. Ershov, A.P. Abstract computability on algebraic structures. In *Algorithms in Modern Mathematics and Computer Science*; Springer: Berlin, Germany, 1981; pp. 397–420.
32. Thompson, S. Axiomatic Recursion Theory and the Continuous Functionals, *J. Symbolic Logic* **1985**, *50*, 442–450.
33. Skordev, D. *Computability in Combinatory Spaces: An Algebraic Generalization of Abstract First Order Computability*; Kluwer Academic Publishers: Dordrecht-Boston-London, Netherlands/USA/UK, 1992.
34. Friedman, H.; Hirst, J. Weak comparability of well orderings and reverse mathematics. *Ann. Pure Appl. Logic* **1990**, *47*, 11–29.
35. Giusto, M. Simpson S.G. Located Sets and Reverse Mathematics. *J. Symbolic Logic* **2000**, *65*, 1451–1480.
36. Friedman, H. Systems of second order arithmetic with restricted induction, I, II (Abstracts). *J. Symbolic Logic* **1976**, *41*, 557–559.
37. Simpson, S.G. Subsystems of Second Order Arithmetic. In *Perspectives in Mathematical Logic*; Springer-Verlag: Berlin, Germany, 1999.
38. Fraenkel, A.A.; Bar-Hillel, Y. *Foundations of Set Theory*; North-Holland: Amsterdam, The Netherlands, 1958.
39. Kurosh, A.G. *Lectures on General Algebra*; Chelsea Publishing Company: New York, NY, USA, 1963.
40. Kelley, J.L. The Tychonoff product theorem implies the axiom of choice. *Fund. Math.* **1950**, *37*, 75–76.
41. Howard, P.; Rubin, J. *Consequences of the Axiom of Choice*. In *Mathematical Surveys and Monographs*; American Mathematical Society: Boston, MA, USA, 1998.
42. Kolmogorov, A.N.; Fomin, S.V. *Elements of the Theory of Functions and Functional Analysis*; Dover Publications: New York, NY, USA, 1999.
43. Bernays, P.; Hilbert, D. In *The Encyclopedia of Philosophy,* Version 3; Macmillan and The Free Press: New York, NY, USA, 1967; pp. 496–504.
44. Boker, U.; Dershowitz, N.A. Formalization of the Church-Turing Thesis for State-Transition Models, 2004. Available online: http://www.cs.tau.ac.il/, accessed on 18 September 2012.
45. Dershowitz, N.; Gurevich, Y.A. Natural Axiomatization of Computability and Proof of Church's Thesis. *Bull. Symbolic Logic* **2008**, *14*, 299–350.
46. Burgin, M. Mathematical Theory of Information Technology. In Proceedings of the 8th WSEAS International Conference on Data Networks, Communications, Computers (DNCOCO'09); Baltimore, MD, USA, 2009; pp. 42–47.

47. Burgin, M. Levels of System Functioning Description: From Algorithm to Program to Technology. In Proceedings of the Business and Industry Simulation Symposium, FL, USA, 2003; pp. 3–7.
48. Dodig-Crnkovic, G.; Müller, V. A Dialogue Concerning Two World Systems: InfoComputational *vs.* Mechanistic. In *Information and Computation*. Dodig-Crnkovic, G., Burgin, M., Eds.; World Scientific: Singapore, 2011.
49. Stepney, S.; Braunstein, S.L.; Clark, J.A.; Tyrrell, A.M.; Adamatzky, A.; Smith, R.E.; Addis, T.; Johnson, C.G.; Timmis, J.; Welch, P.H.; *et al*. Journeys in non-classical computation I: A grand challenge for computing research. *Int. J. Parallel Emergent Distrib. Syst.* **2005**, *20*, 5–19.
50. Cooper, S.B. The Mathematician's Bias- and the Return to Embodied Computation. In *A Computable Universe.* In *Understanding Computation & Exploring Nature As Computation*, Zenil, H., Ed.; World Scientific: Singapore, 2012.
51. Dodig-Crnkovic, G. Significance of Models of Computation from Turing Model to Natural Computation. *Mind. Mach.* **2011**, *21*, 301–322.
52. Burgin, M.; Dodig-Crnkovic, G. From the Closed Classical Algorithmic Universe to an Open World of Algorithmic Constellations. In *Computing Nature,* Dodig-Crnkovic, G., Giovagnoli, R., Eds.; Springer: Heidelberg, Germany, SAPERE series, 2012.
53. Ziegler, M. Physically-relativized Church-Turing Hypotheses. *Appl. Math. Comput*. **2009**, *215*, 1431–1447.