Algorithms: From Al-Khwarizmi to Turing and Beyond

Wolfgang Thomas

Abstract The foundational work of Alan Turing and contemporaries on computability marked a turning point in the development of mathematical sciences: It clarified in a rather absolute sense what is computable in the setting of symbolic computation, and it also opened the way to computer science where the use of algorithms and the discussion on their nature was enriched by many new facets. The present essay is an attempt to address both aspects: We review the historical development of the concept of algorithm up to Turing, emphasizing the essential role that logic played in this context, and we discuss the subsequent widening of understanding of "algorithm" and related "machines", much in the spirit of Turing whose visions we see realized today.

Keywords Algorithm • Al-Khwarizmi • Computability • Logic • Leibniz • Turing

1 Prologue

Alan Turing's contributions to science and engineering are extraordinary, both in depth and in breadth. His work spans scientific fields as represented in the four volumes of his Collected Works (pure mathematics, mathematical logic, machine intelligence, and morphogenesis), and it also covers less well documented work in "engineering" disciplines, such as computer architecture.[1] The present paper focuses on a very specific but most prominent aspect of Turing's heritage, namely his analysis of symbolic computation and the perspectives he connected with the idea of "machines" for intellectual tasks. The pivot element in this discussion

[1]On the occasion of the Alan Turing Year 2012, a new presentation of Turing's work, including hitherto unpublished papers, is available in the volume [5].

W. Thomas (✉)
RWTH Aachen, Lehrstuhl Informatik 7, 52056 Aachen, Germany
e-mail: thomas@informatik.rwth-aachen.de

G. Sommaruga, T. Strahm (eds.), *Turing's Revolution*,
DOI 10.1007/978-3-319-22156-4_2

is his pioneering paper “On computable numbers, with an application to the Entscheidungsproblem” of 1936 [24].

In a first part we describe the central role that logic played in the long process of clarification of the concept of “algorithm” that culminated in the work of Turing and his contemporaries Church, Kleene, and Post. This role of logic is worth being emphasized since “algorithms” were originally understood as procedures for numeric calculation. In the second part we discuss the development towards today’s systems of information processing that has led to a much more comprehensive view on “algorithms”, not just in scientific discourse but much more so in cultural and political discussions in the wider public.

This paper offers observations that cannot be called original; and also the historical sketch we provide is rough and does not touch a number of stages that may also be considered relevant.[2] Nevertheless, we found it worth trying to develop the larger picture around the idea of algorithm and—in terms of time—to address an extended historical axis, centered at 1936, the year when Turing’s landmark paper appeared.

2 Some Prehistory: Al-Khwarizmi and Leibniz

In the work of Turing and his contemporaries, the terms “procedure”, “finite process”, and (as mostly used by Turing) “machine” occur more often than “algorithm”. All these terms, however, point to the same idea: a process of symbolic computation fixed by an unambiguous and finite description.

The word “algorithm” originates in the medieval “algorism” as a recipe to perform calculations with numbers, originally just natural numbers. “Algorism” goes back to one of the most brilliant scientists of the islamic culture, Al-Khwarizmi (around 780–850), who worked in the “House of Wisdom” of the Chalif of Bagdad.[3] In this academy, founded by the Chalif Harun Al-Rashid and brought to culmination by his son Al-Mamun, scientists were employed for a wide spectrum of activities, among them translations (e.g. from Greek and Persian to Arabic), construction of scientific instruments, expeditions, and—quite important—advice to the Chalif. Al-Khwarizmi must have been a leading member. His full name (adding together all name ingredients we know of) was Muhammad Abu-Abdullah Abu-Jafar ibn Musa Al-Khwarizmi Al-Majusi Al-Qutrubbulli. The attribute “Al-Khwarizmi” points to the province of “Choresmia”, located in today’s Usbekistan, where he probably was born and grew up. He was sent by the Caliph to Egypt for an exploration the giza pyramids, he undertook measurements (e.g., executing the experiment of Eratosthenes to determine the diameter of the earth), and he wrote treatises.[4]

[2] Among the more comprehensive sources we mention [6].

[3] For an interesting account on the “House of Wisdom”, we recommend [11].

[4] For a more detailed summary of Al Khwarizmi’s life see, e.g., [27].

The most influential ones were his book on algebra (“Kitāb al-mukhtasar fi hisab al-jabr wa’l-muqabala”) and his text “Computing with the Indian Numbers” (“Kitāb al-Jamʿ wa-l-tafrīq bi-ḥisāb al-Hind”). We concentrate here on the latter, in which he describes the execution of the basic operations of arithmetic (addition, multiplication, and others) in the decimal number system. The Indian sources he used are not known. Also the original text of Al-Khwarizmi seems to be lost. We have translations into Latin, for example the famous manuscript of the thirteenth century kept at the library of the University of Cambridge (England).[5] This text, however, is a bad example of scientific literature: Citations and comments are mixed into a conglomerate, and also many places where the decimal ciphers should appear remain empty. Probably the monk who wrote this text was eager to put everything into a solid theological context, and he was not comfortable with writing down these strange decimal symbols. Thus one has to guess at several places how the missing example computations would look like that should clarify the textual descriptions. It is amusing to read the phrase “but now let us return to the book”, indicating that the author comes back to Al-Khwarizmi. And thus, many paragraphs start with the repetitive phrase “Dixit Algorizmi”—which motivated the term “algorism” for the procedures described in this work.

It is noteworthy that this concept of “algorithm” clearly refers to a process of symbol manipulation, in contrast to calculations performed on the abacus. The arrangement of pieces on the abacus also reflects the decimal system, but the computation process there is not symbolic in the proper sense of the word.

A new dimension to symbolic computation was added by Gottfried Wilhelm Leibniz (1646–1716). Extending ideas of precursors (among them Ramon Llull and Anastasius Kircher), he developed the vision of calculating truths (true statements) and not just numerical values. This vision was partly motivated by the fierce theological disputes of his time, a phenomenon which was not just academic but penetrated politics. Leibniz was born at the very end of the 30 years’ war that had devastated Germany and that was partly rooted in theological conflicts between the catholic and the protestant. Leibniz dreamed of a universal calculus that would help philosophers in their disputes by just following the call “Calculemus!” He hints at his concept of a “characteristica universalis” in a letter to Duke Johann Friedrich of Braunschweig-Lüneburg [6]:

> In philosophy I found some means to do, what Descartes und others did via Algebra and Analysis in Arithmetic and Geometry, in all sciences by a combinatorial calculus [“per

[5] A full presentation in facsimile with transcription to Latin is given in [26]; a translation to English in [2].

[6] Leibniz wrote this letter [13] of 1671 to a duke and not to a colleague; hence he used German rather than Latin, with some Latin words inserted: “In Philosophia habe ich ein Mittel funden, dasjenige was Cartesius und andere per Algebram et Analysin in Arithmetica et Geometria gethan, in allen scientien zuwege zu bringen per Artem Combinatoriam [...]. Dadurch alle Notiones compositae der ganzen welt in wenig simplices als deren Alphabet reduciret, und aus solches alphabets combination wiederumb alle dinge, samt ihren theorematibus, und was nur von ihnen zu inventiren müglich, ordinata methodo, mit der zeit zu finden, ein weg gebahnet wird.”

Artem Combinatoriam"] [. . .]. By this, all composed notions of the world are reduced to few simple parts as their Alphabet, and from the combination of such alphabet [letters] a way is opened to find again all things, including their truths ["theorematibus"], and whatever can be found about them, with a systematic method in due time.

Leibniz undertook only small steps in this huge project, but in a methodological sense he was very clear about the task. As he suggests, logic should be applied by procedures of "alphabet's combination", i.e., symbolic computation. And he was very definite about his proposal to join the algorithmic procedures known from arithmetic with logic. This idea of "arithmetization of logic" (which later Hilbert pursued in his program to show the consistency of mathematics) is raised in two ways:

In his paper "Non inelegans specimen demonstrandi in abstractis" of 1685 [15] ("A not inelegant example of abstract proof method"), he develops the rudiments of Boolean algebra, using equations such as "$\mathrm{A} + \mathrm{A} = \mathrm{A}$" with "$+$" as a sign for union. As an example, let us state the last theorem (XIII) of his note:

Si coincidentibus addendo alia fiant coincidentia, addita sunt inter se communicantia

i.e.,

If from two equal entities we get, by adjoining something, other but again equal entities, then among the added parts there must be something in common

or in a more formal set theoretic terminology, using the symbols of Leibniz:

If to A we add B, respectively N, and we obtain again "coincidentia", i.e. $\mathrm{A} + \mathrm{B} = \mathrm{A} + \mathrm{N}$, and these are "alia", i.e. proper supersets of A, then B and N must have a nonempty intersection.

Clearly this approach to reasoning prepares Boolean algebra as a calculus, using notation of arithmetic.

The second idea on the arithmetization of logic appears in his note "Elementa calculi" (Elements of a calculus) of 1679 [16], where we find the following passage[7]:

For example, since man is a rational animal (and since gold is the heaviest metal), if hence the number for animal (for metal) is a (such as 2) (m such as 3) and of rational (heaviest) r such as 3 (p such as 5), then the number for man or h will be the same as ar, which in our example is 2×3, i.e. 6 (and the number for gold, s, will be the same as mp, which in this example is 3×5, i.e. 15).

We see very clearly the idea to represent elementary concepts by prime numbers and their conjunction by products of prime numbers, which allows to reestablish the factors. This prepares the idea of Gödel numbering that entered the stage again 250

[7]"Verbi gratia quia Homo est Animal rationale (et quia Aurum est metallum ponderosissimum) hinc si sit Animalis (metalii) numerus a ut 2 (m ut 3) Rationalis (ponderosissimi) vero numerus r ut 3 (p ut 5) erit numerus hominis seu h idem quot ar id est in hoc exemplo 2,3 seu 6 (et numerus auri solis s idem quot mp id est in hoc exemplo 3,5 seu 15."

years later—using number theoretic facts to code complex objects (like statements or proofs) by numbers—in a way that allows unique decomposition.

It is somewhat breathtaking to see how optimistic Leibniz was about the realization of his ideas. In a note[8] of 1677 he writes

> When this language is introduced sometime by the missionaries, then the true religion which is unified to the best with rationality, will be founded firmly, and one does not need to fear a renunciation of man from it in the future, just as one does not need to fear a renunciation from algebra and geometry.

This idea of a rational theory of ethics was shared by many of Leibniz's contemporaries. As examples we just mention Spinoza's treatise "Ethica. Ordine geometrio demonstrata" (1677) and the dissertation "Philosophia practica universalis, methodo mathematica conscripta" (1703) of Leibniz's student Christian Wolff.

But more than his colleagues, Leibniz formulated rather bold promises—in a very similar way as we do today when we apply for project money[9]:

> I think that some selected people can do the job in five years, and that already after two years they will reach a stage where the theories needed most urgently for life, i.e., moral and metaphysics, are manageable by an unfallible calculus.

3 Towards Hilbert's Entscheidungsproblem

Leibniz's dream in its full generality remained (and remains) unrealized. Surprisingly, however, it was materialized in the domain of mathematics. This process started with George Boole who developed the vague sketch of Leibniz into a proper theory: "Boolean algebra". The breakthrough in devising a universal scientific calculus was then achieved by Gottlob Frege. His "Begriffsschrift" (1879) introduces a formal language in which mathematical statements can be expressed, the essential innovation being a clarification of the role of quantifiers and quantification. His own work on the foundations of arithmetic, and in particular the subsequent enormous effort undertaken by Russell and Whitehead in their "Principia Mathematica", opened a way to capture mathematics in a formal system.

But in this development the objectives connected with formalization shifted dramatically. The objective was no longer the Leibnizian approach to compute truths needed in life (or just in mathematics) but of a more methodological nature. The shift occurred with the discovery of contradictions ("paradoxes") in Frege's system. The most prominent problem was the contradiction found independently by Russell and

---

[8]From [14]: "Nam ubi semel a Missionariis haec lingua introduce poterit, religio vera quae maxime rationi consentanea est, stabilia erit et non magis in posterum metuanda erit Apostasia, quam ne hominess Arithmeticam et Geometriam, quam semel dedicere, mox damnent."

[9]From [14]: "Aliquot selectos homines rem intra quinquennium absolvere posse puto; intra biennium autem doctrinas, magis in vita frequentalas, id est Moralem et Metaphysicam, irrefragabile calculo exhibebunt."

Zermelo inherent in the concept of a “set of those sets that do not contain themselves as elements”. The formalization of mathematics was now pursued as a way to show its consistency. As Hilbert formulated in his program on the foundations of mathematics, the task was to analyze the combinatorial processes in formal proofs and by such an analysis arrive at the result that the arithmetical equation $0 = 1$, for example, cannot be derived. In pursuing this program, the key issues were axiomatizations of theories, the consistency of theories, and the soundness and completeness of proof calculi.

The fundamental results of Gödel (completeness of the first-order proof calculus and incompleteness of any axiomatic system of arithmetic) made it clear that only in a fragmentary way there was hope to fulfill Hilbert’s program. An essential ingredient in Gödel’s approach was the arithmetization of logic (today called “Gödelization”), transforming Leibniz’s hint mentioned above into a powerful method.

However, a positive result of the foundational research of the early twentieth century was that the “atomic ingredients” of mathematical proofs, as condensed in the rules of the proof calculus of first-order logic, were established. Together with the axiomatization of set theory, a framework emerged in which most of mathematics could be formally simulated. This framework clarified to a large extent which kind of symbolic manipulations are necessary to do logic algorithmically—as Al-Khwarizmi had explained this centuries before for numeric calculations. Most remarkably, it was an algorithmic problem in the domain of logic (and not in the domain of arithmetic) which motivated a general analysis of computability and hence of “algorithm”.

This was “Hilbert’s Entscheidungsproblem”, as formulated in the monograph “Einführung in die theoretische Logik” by Hilbert and Ackermann (1928).

> Das Entscheidungsproblem ist gelöst, wenn man ein Verfahren kennt, das bei einem vorgelegten logischen Ausdruck durch endlich viele Operationen die Entscheidung über die Allgemeingültigkeit bzw. Erfüllbarkeit erlaubt.

Turing’s paper “On computable numbers, with an application to the Entscheidungsproblem” [24] solves this problem in the negative, and it does so by a radical reduction of “algorithm” to very elementary steps of symbol manipulation.

Before discussing this work in more detail, let us emphasize again that the objectives of Hilbert and his colleagues were quite distinct from the visions that Leibniz had in mind, although one might say that axiomatic set theory is the fulfillment of Leibniz’s project of devising a “characteristica universalis”, restricted to the field of mathematics. Rather than studying global properties of formal systems, such as consistency and completeness, Leibniz wanted to use formal rules as a “knowledge engineer”: to find and verify interesting statements by calculation—primarily in areas far beyond mathematics. Hilbert did—as far as we know—not adopt the view that formalization would help in any way to solve concrete mathematical problems, by performing the algorithmic execution of proofs. For him and most logicians in his tradition, the formalization of mathematics is an approach to understand its methodological foundations.

Only today, in computer science, both traditions of “formal logic” are merged again: In computer science, formal systems are set up in many ways, for example as programming languages or as query languages for data bases, and in this design questions of soundness, completeness, and complexity have to be addressed. But we see at the same time the application of these formal systems of data processing to solve concrete problems in virtually all sciences and domains of human life, very much in the spirit of Leibniz.

4 Turing’s Breakthrough

Turing learned about Hilbert’s Entscheidungsproblem in lectures of Max Newman in Cambridge, after 4 years of (very successful) studies of mathematics. It was the fresh look of a young genius that helped to settle the problem. The paper he wrote is one of the most remarkable documents of mathematical literature of the twentieth century. In fact, the solution of the Entscheidungsproblem (which was solved independently by Alonzo Church [3]) is only one of at least seven innovations which Turing offered in his paper:

1. A machine model capturing computability
2. Its justification
3. Conception and implementation of a universal program
4. Establishment of a non-solvable problem
5. Proof that Hilbert’s Entscheidungsproblem is undecidable
6. Equivalence between Turing machines and λ-calculus
7. Initial steps to computable analysis

We do not repeat here the precise formulation of the model of Turing machine. It should be noted that a variant of this model (“finite combinatory processes”) was presented in the same year 1936 by Emil Post [17]. What makes Turing’s paper so brilliant is the mature and conceptually tight justification of his model. This justification starts with phrases which remind us precisely of the algorithms that were the subject of Al-Khwarizmi:

> Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child’s arithmetic book . . . .

A second remarkable aspect in Turing’s paper is the fact that after presenting his model of Turing machine, he immediately exhibits a problem that is not solvable with this model. For this, he develops the idea of a universal machine, enters the technicalities of actually constructing one (and, as an aside, introduces the programming technique today called “macros” for this purpose), and then applies a diagonalization argument. This appearance of a powerful model connected immediately with a corresponding unsolvability result should be compared with the centuries that elapsed between the clear understanding of algebraic expressions (in

Vieta's time) and the proof of Abel that for polynomials of degree 5 one cannot in general find solutions in this format.

In fact, the mere possibility to envisage algorithmically unsolvable problems emerged only at a rather late stage. In 1900, in the formulation of Hilbert's 10th problem

> Eine diophantische Gleichung mit irgendwelchen Unbekannten und mit ganzen rationalen Zahlenkoeffizienten sei vorgelegt: Man soll ein Verfahren angeben, nach welchem sich mittels einer endlichen Anzahl von Operationen entscheiden lässt, ob die Gleichung in ganzen rationalen Zahlen lösbar ist.

One just finds the task to develop a "procedure" ("Verfahren"). The earliest place in mathematical literature where the certainty about algorithmic solutions is put into doubt seems to be a most remarkable paper by Axel Thue of 1910 ("Die Lösung eines Spezialfalles eines allgemeinen logischen Problems" [23]). He formulates the fundamental problem of term rewriting: Given two terms s, t and a set of axioms as equations between terms, decide whether from s one can obtain t by a finite number of applications of the axioms. He resorts to a special case in order to provide a partial solution. About the general case one finds the following prophetic remark:

> A solution of this problem in the most general case might perhaps be connected with unsurmountable difficulties.[10]

It is a pity that this brilliant paper remained unnoticed for decades; one reason for this is perhaps its completely uninformative title. (A detailed discussion is given in [21].)

The work of Turing and his contemporaries Church, Kleene, Post finished a struggle of many centuries for an understanding of "algorithm" and its horizon of applicability, termed "computability". This success was possible by a merge of two traditions in symbolic computation: arithmetic and logic. The impression that an unquestionable final point was reached with this work was underlined by Gödel, who stated in 1946, 10 years after Turing's breakthrough [7]:

> Tarski has stressed [...] (and I think justly) the great importance of the concept of general recursiveness (or Turing's computability). It seems to me that this importance is largely due to the fact that with this concept one has for the first time succeeded in giving an absolute definition of an interesting epistemological notion, i.e., one not depending on the formalism chosen. [...] By a kind of miracle it is not necessary to distinguish orders.

5 Moves Towards Computer Science

The year 1936 not only marks a point of final achievement but is at the same time the initialization of a new and rapidly developing discipline: computer science (or

[10]"Eine Lösung dieser Aufgabe im allgemeinsten Falle dürfte vielleicht mit unüberwindlichen Schwierigkeiten verbunden sein."

"informatics"). Each of the pioneers mentioned above in connection with Turing, namely Church, Kleene, and Post, as well es Turing himself, were active in this launch of a new scientific subject.

Turing himself turned, for example, to questions that are hosted today in the field of "computer architecture", but he also addressed many further issues, such as program verification. In 1957, Church formulated a fundamental problem beyond program verification—"Application of recursive arithmetic to the problem of circuit synthesis" [4]—thus opening a fascinating branch of computer science, in which today game theoretic methods are used for the automatic construction of interactive programs (see, e.g. [22]). Kleene should be noted for his path-breaking work [12] on regular events and finite automata, establishing the basic equivalence result in automata theory. Finally, Post was the first to exhibit purely combinatorial (and thus purely "mathematical", as opposed to logical) undecidable problems (among them Post's Correspondence Problem [19]), and he developed in [18] a theory of problem reduction that today underlies much of complexity theory.

The subsequent rise of computer science changed our views on algorithms in two ways: First, algorithmic methods in computer science transcend the framework of symbol manipulation inherent in Turing's analysis. Secondly, "algorithms" are understood today in the context of the highly complex software systems that govern our life (e.g., in enterprise software, internet applications, etc.) Let us address them both.

6 New Facets of "Algorithm"

Turing's analysis (as well as the parallel work of Post) refers to procedures that work on finite words composed from symbols taken from a finite alphabet. As noted by Turing, the algorithms of basic arithmetic can be treated in this framework. However, a standard first-year course in computer science, usually titled "Algorithms and data structures", already shows several chapters that go beyond this domain. In particular, we deal there with algorithms over trees and over graphs rather than words. In this generalized setting, some features of algorithms arise that are hidden when we work over words. For example, over graphs we observe the lack of a natural ordering (e.g. for the set of vertices). This lack of order allows to say "Pick an element in the set V of vertices ..." without the requirement (and indeed, without the possibility) of fixing a particular element. Over words, the situation is different: Picking a letter in a word always implies the possibility to pick a particular (e.g., the first) letter. As Gurevich and others remarked, the Turing model working with the substrate of words on a tape does not allow us to deal with algorithms on the adequate level of abstraction. The machinery of coding (by words) that enables us to make a bridge to Turing's model spoils this adequacy. To settle this problem, a generalized view on algorithms was developed in the model of "abstract state machine" [8]. It has the flexibility that is needed to answer the challenge of very diverse kinds of algorithms as they are specified, for example, in programming languages.

In some domains of algorithmic mathematics, a more abstract view (than that of computations over words) is unavoidable even when allowing “codings by words”. An example is the domain of classical Euclidean geometry, where algorithms in the form of “geometric constructions” are familiar since antiquity. The data handled by these constructions (points, lines, etc.), are infinite objects, as are the real numbers. Points of Euclidean space and real numbers cannot serve as “inputs” to an algorithm in Turing’s sense, since in their generality they are not codable by finite words. Abstract state machines over the space of (tuples of) reals can be invoked to handle geometric algorithms and algorithms over the reals; for the latter, also the Blum-Shub-Smale model [1] provides an adequate framework.

Apart from this, a host of new algorithmic concepts was developed in computer science, often termed as “schemes”, “processes”, etc., which are not immediately covered by the model of Turing machine but constitute clearly mechanisms to transform data according to finite recipes. One can give numerous examples: Classification procedures for image and speech processing (as needed in visual computing, data mining, and automatic speech translation), distributed algorithms that guarantee a convergence of network states to prescribed equilibria, or algorithms for planning and search as used in robotics. A field of growing importance is the use of algorithms in non-terminating reactive systems (such as communication protocols or control systems); here the idea of Turing of a nonterminating process (e.g., to generate the decimal expansion of a real number) is enhanced by the feature that an infinite computation may be subject to a non-termining sequence of receipts of inputs while it works.

These procedures are far away from the simple set-up of symbolic computation considered by Turing in his paper of 1936. But it was exactly Turing who was aware of the perspective of a widening of the horizon of algorithmic methods—at a very early stage. His term was “machine”, allowing him to cover the software and hardware aspects simultaneously. Let us document this by two citations:

> We may hope that machines will eventually compete with men in all purely intellectual fields. [25]
>
> One way to setting about our task of building a “thinking machine” would be to take a man as a whole and try to replace all parts of him by machinery. This would include television cameras, microphones, loudspeakers, wheels, and “handling servo-mechanisms”, as well as some sort of “electronic brain”.[11]

[11]Cited from [10, p. 117].

7 Algorithms as Molecules in Large Organisms

The vision of "thinking machine" as sketched by Turing is a reality today. Such systems are developed in the field of robotics; they are serious approximations of living organisms.

Indeed, computer science has created hierarchies and networks of algorithms of such a huge complexity that their description necessarily makes use of terms familiar from biology. "Life cycle", "software evolution", "virus", "worm", "infection" are common terms in the discussion of large software systems or networks. One may say that the instructions for Turing machines represent the atomic level of data processing, and algorithms (in the classical sense) the molecular level. Based on this, much larger systems are designed in which the global structure and sensory elements for the interaction with the environment (e.g., humans) are often more essential features than the individual components.

A methodological problem of computer science is the span of orders of magnitude realized in the huge data conglomerates and software systems as we see them today. If we consider the physical world, start from the atoms as basic units, and proceed in stages to larger physical objects, where one step involves an enlargement by a factor 1000, we reach, successively in four steps, objects of the following kind: a molecule, a living cell, a small animal, a small village. These kinds of objects are studied in separate scientific disciplines, ranging from physics via chemistry and biology to sociology. Each of these disciplines has specific—and rather different—models that are suitable for the respective objects under consideration. In computer science we may start with a byte corresponding to the atomic level (describing, for example, an alphanumeric symbol as handled in a Turing machine instruction), and proceed in four steps to a Kilobyte (corresponding to a small program of a dozen lines), a Megabyte (corresponding to a book, or to code produced by a team of students during a software project), a Gigabyte (a size of data comparable to the Windows operating system), and a Terabyte (comparable to the stock of the Library of Congress). Much larger are the "big data" handled in the world wide web. Although this huge span of orders of magnitude is studied in one science, computer science, it is obvious that rather diverse methodologies (as in the natural sciences) are needed for an adequate study. A common misunderstanding of the role of Turing machines is the remark that this model "does not match the reality of computer science systems today". This remark is as misplaced as is the rather empty statement that atomic physics does not capture the reality of living organisms.

Despite this richness of the landscape of computer systems and computer science, the public view on the intelligent machinery designed by computer scientists puts the term "algorithm" at its center, most often in the plural form. "Algorithms" is what make devices and processes of information technology run. Any conglomerate of units of information processes is covered by this term. This is different from the situation in biology where one does not say that biological systems are "molecules".

Let us illustrate this observation on the colloquial use of "algorithms" by three headlines the author noted in 2012–2013, taken from the press, respectively the web.

The deeply troubling perspective of programmed robots designed for military combat (on earth and on air) was discussed with the subtitle “The moral of algorithms” in a leading German newspaper.[12] In connection with the comprehensive analysis of data on the web (covering millions of persons) by government agencies, the term “the tyranny of algorithms” was used in an article of the same newspaper,[13] and, finally, the controversy in this discussion was condensed by “Spiegel Online” into the remarkable headline “Freedom against algorithms”.[14]

While it is clear to the experts that current implementations of computer systems ultimately rely on small computation steps in microprocessors and are thus in principle reducible to Turing machine computations, we see that in the public discussion the actual understanding of “algorithms” drastically exceeds the content of the Turing model—it is today located on a much more general level.

8 Returning to Leibnizian Visions?

The current colloquial usage of “algorithms” seems approaching the visions of Leibniz on treating central questions of human behavior and social disputes by calculation. Indeed, in an unexpected sense, our contemporary information processing systems are coming close to Leibniz’s idea—envisaging the solution of moral, social, and political questions by algorithms. We observe today that software systems used in the financial markets and in military and government institutions are installed precisely for the purpose of decision finding in questions of economics, politics, and even war.[15]

Thus, “algorithms” in this general view, i.e., the algorithmic abilities of the most sophisticated software and hardware systems that computer scientists develop today, silently put into reality a fair amount of Leibnizian utopia. But in contrast to Leibniz’s hopes, it seems that the “unfallible calculus” underlying these systems can no more be seen as a secure means to get nearer to the “best of all worlds”; rather we are confronted with new troubling questions on estimating the power and justifying the use of algorithms.

[12]F. Rieger, Das Gesicht unserer Gegner von morgen, Frankfurter Allgemeine Zeitung, 20th Sept. 2012.

[13]G. Baum, Wacht auf, es geht um die Menschenwürde, Frankfurter Allgemeine Zeitung, 16th June 2013.

[14]“Freiheit gegen Algorithmen”, Spiegel Online, 21st June 2013.

[15]This aspect, with a focus on the role of algorithmic game theory, is developed at length by F. Schirrmacher, a leading German journalist, in [20], a bestseller on the German book market.

References

1. L. Blum, M. Shub, S. Smale, On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. Bull. Am. Math. Soc. **21**, 1–46 (1989)
2. J.N. Crossley, A.S. Henry, Thus spake al-Khwārizmī: a translation of the text of Cambridge University Library Ms. Ii.vi.5. Hist. Math. **17**, 103–131 (1990)
3. A. Church, A note on the Entscheidungsproblem. J. Symb. Log. **1**, 40–41 (1936)
4. A. Church, in: *Summaries of the Summer Institute of Symbolic Logic*. Application of recursive arithmetic to the problem of circuit synthesis, vol. I (Cornell University, Ithaca, 1957), pp. 3–50
5. B. Cooper, J.V. Leeuwen (eds.), *Alan Turing: His Work and Impact* (Elsevier, Amsterdam, 2013)
6. M. Davis, *The Universal Computer – The Road from Leibniz to Turing*. Turing Centennial Edition (CRC Press, Boca Raton, 2012)
7. K. Gödel, Remarks before the Princeton bicentennial conference on problems in mathematics, in *Kurt Gödel, Collected Works*, ed. by S. Feferman et al., vol. II (Oxford University Press, Oxford, 1990), pp. 150–153
8. Y. Gurevich, Sequential abstract-state machines capture sequential algorithms. ACM Trans. Comput. Log. **1**, 77–111 (2000)
9. H. Herring (ed.), *G.W. Leibniz Schriften zur Logik und zur philosophischen Grundlegung von Mathematik und Naturwissenschaft* (lat. u. deutsch) (Suhrkamp, Frankfurt, 1996)
10. A. Hodges, *Alan Turing: The Enigma* (Vintage, London, 1992)
11. J. Al-Khalili, *The House of Wisdom: How Arabic Science Saved Ancient Knowledge and Gave Us the Renaissance* (Penguin Press, New York, 2011)
12. S.C. Kleene, Representation of events in nerve nets and finite automata, in *Automata Studies*, ed. by C.E. Shannon, J. McCarthy (Princeton University Press, Princeton, 1956), pp. 3–41
13. G.W. Leibniz, Brief an Herzog Johann Friedrich von Braunschweig-Lüneburg (Okt. 1671), in *Philosophische Schriften von Gottfried Wilhelm Leibniz*, ed. by C.I. Gerhardt, vol. 1 (Weidmannsche Buchhandlung, Berlin, 1875), pp. 57–58
14. G.W. Leibniz, Anfangsgründe einer allgemeinen Charakteristik, in [9], pp. 39–57
15. G.W. Leibniz, Ein nicht unelegantes Beispiel abstrakter Beweisführung, in [9], pp. 153–177
16. G.W. Leibniz, Elemente eines Kalküls, in [9], pp. 67–91
17. E.L. Post, Finite combinatory processes – formulation 1. J. Symb. Log. **1**, 103–105 (1936)
18. E.L. Post, Recursively enumerable sets of positive integers and their decision problems. Bull. Am.. Math. Soc. **50**, 284–316 (1944)
19. E.L. Post, A variant of a recursively unsolvable problem. Bull. Am. Math. Soc. **52**, 264–268 (1946)
20. F. Schirrmacher, *EGO: Das Spiel des Lebens* (Karl Blessing-Verlag, München, 2013)
21. M. Steinby, W. Thomas, Trees and term rewriting in 1910: on a paper by Axel Thue. Bull. Eur. Assoc. Theor. Comput. Sci. **72**, 256–269 (2000)
22. W. Thomas. Infinite games and verification, in *Proceedings of International Conference on Computer Aided Verification CAV'02*. Lecture Notes in Computer Science, vol. 2404 (Springer, Berlin, Heidelberg, New York, 2002), pp. 58–64
23. A. Thue, Über die Lösung eines Spezialfalls eines allgemeinen logischen problems. *Kristiania Videnskabs-Selskabets Skrifter*. I. Mat. Nat. Kl. 1910, No. 8
24. A.M. Turing, On computable numbers, with an application to the Entscheidungsproblem. Proc. Lond. Math. Soc. **42**, 230–265 (1936)
25. A.M. Turing, Computing machinery and intelligence. Mind **59**, 433–460 (1950)

26. K. Vogel, *Mohammed ibn Musa Alchwarizmi's Algorismus. Das früheste Lehrbuch zum Rechnen mit indischen Ziffern* (Zeller, Aalen, 1963)
27. H. Zemanek, Dixit algorizmi: his background, his personality, his work, and his influence, in *Algorithms in Modern Mathematics and Computer Science*, ed by A Ershov, D Knuth. Proceedings, Urgench, Uzbek SSR, 16–22 September 1979. Springer Lecture Notes in Computer Science, vol. 122 (Springer, Berlin, 1981), pp. 1–81