A personal account of Turing's imprint on the development of computer science

J. Díaz* C. Torras †

Abstract

The first part of the XX century saw the development of the digital computer and the field of computer science. In the present paper, we sketch our vision of that period and of the role that Alan Turing and some of his contemporary peers played in that development.

1 Introduction

This year there have been more than 100 celebrations honoring the birth of Turing, the size of which contrasts a bit with the celebrations devoted to other pioneers of the computability field, as Kurt Gödel or John von Neumann, whose anniversaries were remembered with specialized meetings in a few universities, which did not reach by far the popularity of the present year celebrations.

In today's academic world, it is normal that scientists publish mainly joint papers and tend to concentrate their research effort in a few, very focused, interconnected topics. This social trend of research is a rather new one. Alan M. Turing (1912-1954) was an English mathematician, which in spite of his short life, made contributions in many different fields of mathematics, biology and computer science. For instance, as an undergraduate at Cambridge he re-discovered the central limit theorem, and the proof was sufficiently novel that Cambridge university awarded him a fellowship for Princeton [85]. While at Bletchley Park during the WWII, he applied again probability to decrypt the German ciphers produced by the Enigma machine[1]. For an historical and technical explanation of Turing's efforts in

*Llenguatges i Sistemes Informàtics, UPC; email: diaz@lsi.upc.edu

†Institut de Robòtica i Informàtica Industrial, CSIC-UPC; email: torras@iri.upc.edu

[1]The two technical reports dealing with his probabilistic approaches to break the Enigma have been declassified in May 2012 [73, 74].

breaking the Enigma's code see Chapter 4 in [59] or [39]. After his 1936 breakthrough result on a virtual model of computer, the *universal Turing machine* (UTM), which will be one of the central topics in the present article, Turing approached an array of problems in mathematics and biology from the perspective of their computability. He contributed to numerical analysis, in particular matrix decomposition manipulation [76] (see [15] for a survey); the Riemann hypothesis [79] (see [7] for a survey); philosophy [77] (this paper has originated a vast literature within the philosophy community); biology, his celebrated paper on morphogenesis [78] (see [58, 35] for the long-term impact of this paper).

In the present work we focus exclusively on the contribution of Turing to computation and his influence on the development of the digital computer machine. Our main thesis is that Turing's work was a very important link in the chain of people and developments that ended with the birth of the universal computer and the concept of decidability, but it seems plausible that, without him, the digital computer would have developed basically the same, despite Turing's thought that digital computers were a practical version of his universal machine (see pp. 1190 in [33]). On the other hand, he was the first person to formalize an algorithmic viewpoint on different types of mathematical and biological structures and, without doubt, he can be considered the father of theoretical computer science.

For the reader interested in the life and achievements of Alan Turing, we recommend the excellent biography by Andrew Hodges [32].

2 Algorithms and calculating machines

From very old times, humanity had the need to find step-by-step procedures to solve problems associated with agriculture, navigation, astronomy, etc. For instance, computing the area of a triangle, computing the radius of the earth, or computing the square root of an integer (see for example [37, 45])[2]. These procedures are what today we call algorithms. An *algorithm* is a set of step-by-step rules to solve a problem, such that, given an input, the algorithm produces a correct output in finite time. A detailed cooking recipe is an algorithm, where the quality of the output is quantitatively difficult to evaluate. The word algorithm derives from the word algorism, the middle ages latin manner to refer to the rules of performing arithmetic as explained in the textbook "Kitab al-Jam'a wal-Tafreeq bil Hisab al-Hindi",

[2]One of the most interesting source of references for the history of algorithmics are Knuth's books.

the arithmetic book written by mathematician Abu Ja'far Muhammed ibn Mûsâ Al-Khwârizmî [86, 38].

In parallel there was a progressive emphasis on building analog mechanisms to carry out computations. In an *analogue machine* numbers are represented by physical magnitudes. In general, a mathematical process is transformed within these machines into a sequence of movements of gears (or other mechanical system) to obtain the solution.

Examples of known antique analog mechanisms are the astrolabe, the abacus, the slide rule, the nomograph, the pascaline calculator, the Leibniz wheel, etc. A particularly relevant example of old sophisticated analog engine is the *Antikyhera mechanism* (approx. 150 BC), to which some people refer as the first computer. It was recovered from the bottom of the sea in 1900, and until the 21c. it was not clear what it could do; now it is known that it could compute the position and phases of the moon and the sun. It was probably used for sailors to know the tides in the costs and to predict other astronomical phenomena. Its precision is comparable to the one of a 19c. Swiss clock (see [63] and Chapter 3 in [11]).

Until the 20c. scientists used the progressively more sophisticated calculating mechanisms only as a helping tool to implement particular arithmetic steps of algorithms, with the exception of a few visionary minds like Leibniz and Babbage.

3 The dream of a universal method of reasoning

Gottfried W. Leibniz (1646-1716) is known to be an important mathematician, philosopher, jurist, and inventor (among other things he designed a calculator, the *Leibniz wheel*, that could multiply directly by using cogwheels of different radii). However, from the computational point of view, what is interesting is Leibniz's dream of finding an automatic reasoning machine relying on an alphabet of unambiguous symbols[3] manipulated by mechanical rules, which could formalize any consistent linguistic or mathematical system. In the words of Davis (pp. 13 in [18]): "Leibniz saw his program as consisting of three major components. First, before the appropriate symbols could be selected, it would be necessary to create an encyclopedia encompassing the full extent of human knowledge. Once having accomplished this, it should be feasible to select the key underlying notions and to provide appropriate symbols for each of them. Finally, the rules of deduction could be

[3]He picked up the symbols $\int$ and d for the integral and differential calculus.

reduced to manipulations of these symbols, that is what Leibniz called the calculus ratiocinator, which nowadays might be called symbolic logic".

Hence Leibniz believed that human thought and reasoning could be automated. In fact, he thought that his symbolic calculus could be used to settle legal disputes: a few men of good will sitting around a table would say "let us calculate" and decide who was right using paper and pencil or, even better, a machine [18, 24]. It is known that Leibniz knew the *Ars Magna* by Ramon Llull (1232-1315), where he proposed a combination of religious and philosophical arguments taken from previously compiled lists, such that a reader with a question could find an appropriate answer. Llull even designed a kind of machine to implement this, the *Lullian Circles*. Each circle dealt with a particular topic of faith and consisted of two discs inscribed with alphabetical letters or symbols that referred to lists of concepts, where rotating the circles would generate different combinations of arguments. The underlying principle was that there were a limited number of basic, undeniable truths in the catholic faith, and that one could answer any doubt related to that faith by choosing the appropriate combinations of these elemental truths [57, 6].

Besides symbolic calculus, Leibniz also had some ideas that 200 years later played an important role in the development of the digital computer. Although binary numbers were known since the 5th century BC, Leibniz saw binary coding as the key to his universal language. In fact, Leibniz proposed *shift registers* as the way to perform arithmetic operations with binary digits. He also imagined a calculating machine in which binary digits were represented by spherical tokens governed by mechanical control (see Chapter 6 in [24]).

Leibniz did attempt to produce a calculus ratiocinator, and he created an algebra to specify the rules for manipulating logical concepts, with specific symbols. George Boole (1815-1864) and Augustus De Morgan (1806-1871) converted the loose ideas of Leibniz into a formal system, the Boolean algebra. Finally, Gottlob Frege (1848-1925) in his famous *Begriffsschrift*[4] provided the fully developed system of *axiomatic predicate logic*. Later, Frege went on to prove that all the laws of arithmetic could be logically deduced from a set of axioms within his Begriffsschrift. Frege's main innovation was the introduction of variables. He wrote a two-volume book on the formal development of the foundations of arithmetic. When the second volume was already in print, Frege got the celebrated letter from Bertrand Russell

[4]For an English translation, see: *Concept Script, a formal language of pure thought modeled upon that of arithmetic*, in [29] pp. 1879–1931.

(1872-1970) showing his theory was inconsistent. The counterexample was the paradox of *extraordinary* sets. A set is defined to be *extraordinary* if it is a member of itself, otherwise it is called *ordinary*. *Is the set of all ordinary sets also ordinary?* (There are plenty of references for the historical account of the letter and its effects, but see pp. 169 to 171 in [23] for a particularly charming account of the effect of Russell's letter on Frege.)

Frege's work was the spark for 30 years of research to build solid foundations for mathematics. A particularly important piece of work in this research was the book *Principia Mathematica* by Whitehead and Russell [82]. As we will see, the effort ended in 1931 with a negative answer by Gödel. Meanwhile, the end of the 19c. and the beginning of the 20c. witnessed strong mathematical, philosophical and personal battles between the *intuitionist* and *formalist* European schools of logic. Among the many historical accounts of that period, two gentle overviews are [18, 23].

3.1 Human computers

Since the introduction of differential calculus in the 17c., human calculators were hired to perform mechanical computations. For instance, Carl Friedrich Gauss (1777-1855) hired Johann Zacharias Dase, who had an incredible mental capacity to handle large numbers, but could not understand the most basic concepts of mathematics [34]. During the 19c. and the first half of the 20c., the word *computer* meant a human implementing a mathematical procedure, possibly with the help of a calculator or slide rules. Human computers were used in the construction of all kinds of tables: artillery firing tables, nautical and astronomical tables. They played an important role in the Manhattan project to build the atomic bomb. As Grier describes in his book [26], the Mathematical Tables Project employed 450 human computers, mostly people with low incomes, who from 1938 to 1948 produced tables of powers, trigonometric functions, and other mathematical functions to put together the first complete *Handbook of Mathematical Functions*. Each group of workers was assigned a specific task, addition, subtraction, multiplication, or division calculations, making the functioning of a group "similar" to the basic scheme of the future digital computer. The idea of human computers was so deeply embedded into the minds of those who designed the first digital computers, that at the beginning, they denoted the machines *computors* to distinguish them from human computers [24].

3.2 The difference engine and the analytic engine

Charles Babbage (1791–1871) was an English mathematician, cryptographer and inventor. Looking at an astronomic table with the English astronomer John Herschel (1792-1871) they discovered the tables were wrong, which led Babbage to design a machine to compute correctly astronomic data. As he mentioned in [4], the use of human computers in table construction made inevitable the introduction of errors, which were difficult to detect and fix, the only way out being to completely mechanize the process so as to avoid human intervention. Towards this goal, he designed the *Difference Engine* following previous ideas of Johan H. von Müller (1746-1830), a German army engineer who conceived a machine based on Newton's method of divided differences [40], but who never started to build the machine.

Astronomic tables rely heavily on the evaluation of trigonometric functions. The basic idea of the machine is to approximate the value of logarithmic and trigonometric functions by evaluating the polynomial arising in the Taylor's expansion of these functions. For instance, $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$. Therefore, the computation boils down to computing polynomials at a given value x, where the degree of the polynomial depends on the desired accuracy.

Babbage's design to evaluate polynomials up to degree n consisted of $n+3$ columns. The first column contained the values of x, the second, the corresponding values $f(x)$, the third, the values $\Delta_1(x) = f(x+1) - f(x)$, the fourth column contained $\Delta_2(x) = \Delta_1(x+1) - \Delta_1(x)$ until the $n+1$ column contained $\Delta_n(x+1) - \Delta_n(x)$. Note that for any polynomial, of degree n, $\Delta_{n+1}(x) = \Delta_{n+1}(x+1) = \Delta_{n+1}(x+2) = \cdots$. The initial values were computed and introduced by hand into the machine until arriving to the constant value of $\Delta_{n+1}(x)$ at column $(n+3)$; from there on, turning a crank, the machine computed $f(n+1), f(n+2), \ldots$ and output the results in a typewriter. Thus, we only need to introduce by hand the first n values of x and the computations until arriving to $\Delta_{n+1}(0)$, from then we can back-track from column $(n+3)$ to column 1 getting the values from $f(n+1), f(n+2), \ldots$. this only needs sums and subtractions, and thus, it can be done using the cogwheels of the difference engine.

Babbage built parts of the machine but never completed the whole construction. He later designed an improved version, *Difference Engine No. 2*, which never went past the blueprint stage, until in 1991 the British museum built an exemplar of it [66, 67]. A Swedish lawyer, Pehr Scheutz (1785–1873), built several improved versions of the difference engine, based on the ideas of Babbage [40].

While working on the difference engine, Babbage realized he could do a

more general machine. He started to design the *analytic engine*, a mechanical machine with columns of discs, each disc representing a digit between 0 and 9, and each column representing a real number. The important distinction between the difference and the analytic engines is that, while the former was a single purpose calculator, the analytic engine was an *universal analogic mechanism*, which was controlled by an external *program* introduced via punched cards. Moreover, input data was introduced via a different type of punched cards. Thus, the idea of the analytic engine was to have an *arithmetical unit*, which was an improved kind of differential engine able to perform basic arithmetic operations but having also internal procedures to carry out program instructions (it played a role similar to the CPU in the digital computers), and an input/output mechanism by means of punched cards. The whole process was controlled by a program that introduced for the first time the *conditional branching* feature, with control operators like **if then**, **goto**, **for**, etc.

We just aim to give a glimpse of Babbage's contribution. More precise details can be found in [81], where there are plenty of information about the analytical engine, including videos and the original 1842 report of F. Menabrea with the annotations of Ada Augusta Byron, the countess of Lovelace (1815-1852). An English translation of the original can be retrieved from the site devoted to the analytic engine at the *Bibliothèque Universelle de Genève* [44]. The notes from Ada Byron include what is considered to be the first computer program, a procedure to compute the Bernouilli numbers (see note G in [44]). The analytic engine was never constructed, Babbage tried to built some components of it, as the arithmetical unit, but the English government cut the funding and the project was discontinued.

There has been a discussion about the impact of Babbage's ideas on the development of the universal digital computer, with some historians arguing that the analytic engine had no real impact on the development of the digital computer [83, 8]. From today's perspective, after many efforts devoted to the study of the analytic engine, it seems clear that Babbage had the ideas underlying the *modern universal computer*, which is not to say the people involved in the development of computability knew about Babbage's work. On the contrary, evidence seems to indicate that the analytic engine had a minimum influence on the development of the digital computer.

3.3 Babbage's influence

Since in 1836 Gaspar Coriolis designed a mechanical calculator to solve first-order differential equations [14], an increasing number of analogic *differential*

analyzers to solve differential equations of higher degree by integration were proposed. The motivation for that surge of machines was the need for fire-control tables in artillery. However, these machines were considered as single purpose advanced calculators and did not have too much relation with Babbage's universal *analytical engine*. Here we comment on three examples that used some of Babbage's ideas.

- Percy Ludgate (1893-1922) was an Irish accountant that in his spare time designed an analytic engine with remarkable differences with Babbage's one. He did not use gears, but sliding rods and shuttlers, he relied on binary numbers, and probably his greatest contribution was to use a system of multiplication using logarithms, not in a very different way than the slide rule. His design is not easy to follow and never was implemented. For more details see [54], and for a bit cryptic description of his design see [41].

- Leonardo Torres Quevedo (1852-1936) was a Spanish civil engineer that besides constructing a number of cableways (some still in use), dirigibles, and automatic chess players, he also designed several analog calculating machines. He was aware of Babbage's work and in [69] he proposed to build an electromechanical calculator with floating-point arithmetic and using a logarithmic scale. The calculator was controlled by a read-only program with conditional branching (as Ada Lovelace had already proposed for the analytical engine). In 1920 one of his calculators, *the arithmometer*, was demonstrated to the public; people would input an arithmetic problem and the calculator would print the solution (see [54, 68]).

- Horward Aiken (1900-1973) was an US engineer who designed the Harvard Mark I, also know as the Automatic Sequence Controlled Calculator. He explicitly stated that his Mark I machine was the electronic realization of the analytic engine, but it seems Aiken had very limited understanding of how the analytic engine worked [12]. Two differences with the Babbage design are that it was electromechanical and it did not have the conditional branching feature, a big difference with the analytic engine, which would make it difficult to implement some of Ada Lovelace's programs on the Mark I, so the machine was not universal (or *Turing-complete*, as universal machines were denoted for reasons we will see below). Mark I was operational in 1944, and the project was financed by IBM.

4 The unifying theory of computation

David Hilbert (1862–1943) was one of the great mathematical minds in the 20c. Under his leadership, the *Mathematical Institute at Göttingen* became the leading mathematical place in the world, until the 1930's when the Nazi policies displaced the mathematical center of gravity from Göttingen Institute to the *Institute for Advanced Studies* at Princeton. For a nice biography of Hilbert's life see [55].

Since the 1890's, Hilbert had become one of the leading figures in the formalist school trying to solve the foundational crisis of mathematics produced by Russell's letter to Frege. In 1928 Hilbert wrote a textbook with his student Ackerman [31], where he posed the completeness of arithmetic as an open question.

Let us recall some basic definitions. A *formal system* is a language, a finite set of axioms, together with a set of inference rules used to derive expressions (or statements) from the set of axioms. An example is mathematics with the first-order logic developed by Frege. A formal system is said to be *complete* if every statement can be proved or disproved. Otherwise the system is said to be *incomplete*. A formal system is said to be *consistent* if there is not a step-by-step proof within the system that yields a false statement. A system is said to be *decidable* if, there exists a *finite procedure* (i.e., an algorithm) for every statement, to determine whether the statement is true.

At the Paris International Congress of Mathematicians in 1900, Hilbert presented a list of 10 relevant problems to be solved in the 20c. The list was extended to 22 problems in a posterior English written version of the ICM presentation [30]. The 10th problem was: "Given a diophantine equation with any number of unknown variables, devise a process to determine in a finite number of operations whether the equation is solvable". Recall that a diophantine equation is a polynomial equation that allows only integer solutions. For example, find if there are integer solutions to $4xyz = 3(yz + xz + xy)$.

At the Bologna International Congress of Mathematicians in 1928, Hilbert presented three problems to his fellow mathematicians. The second one would have a big impact in the development of computing science, the *Entscheidungsproblem:*[5] "Provide a method such that, given a first-order logic statement, would determine in a finite number of well-defined steps whether or not the statement is valid". In other words, the Entschei-

[5]The translation in English is the decision problem.

dungsproblem asks for the existence of a finite procedure (an algorithm) that could determine if a mathematical statement is true or false. Hilbert was convinced that the answer would be yes, as mathematics should be complete, consistent and decidable.

There were some negative reactions to Hilbert's Entscheidungsproblem; for instance, according to Hodges' book, the English mathematician Godfrey Hardy (1877-1947) that was present at the 1928 ICM made the following comment: "There is of course no such a theorem, and this is very fortunate, since if there were we should have a mechanical set of rules for the solution of all mathematical problems, and our activities as mathematicians would come to an end" (pp. 93 in [32]).

4.1 The incompleteness theorem

Hilbert was born in Königsberg and in 1930 the city honored him with the title of honorable citizen. As part of the festivities, a workshop on the foundations of mathematics was organized. As Davis (pg. 89 in [18]) recounts: "At the round-table discussion that concluded the event, a shy young man named Kurt Gödel made a quiet announcement that, to those who grasped its importance, signaled a new era. One of the presents, was von Neumann[6] who got the point at once, and concluded that the jig was up".

The announcement was that arithmetic was incomplete, i.e., there were statements that could not be either proved or disproved. K. Gödel (1906-1978) was a mathematician with an interesting biography [20]. His celebrated result of 1931 stated that, in any formal system sufficiently powerful to include ordinary arithmetic, there will be undecidable statements, i.e., that can't be proved to be true or false [27, 43]. The proof is technically involved, but the idea is simple. It goes back to the same argument in Russell's letter to Frege. Gödel proved that any arithmetic any formula could be encoded as an integer, so proofs and in fact the whole arithmetics can be encoded as integers. As we are encoding the integers and the arithmetics, it is possible to write arithmetic self-referring statements (something equivalent to Russell's extraordinary sets), which is a sort of paradox that cannot be proved true or false using the rules of arithmetic. For the details of the proof see, for example, the appendix to Ch. 6 in [18] or the nice description

[6] John von Neumann (1903-1057) was an extraordinary mathematician that made relevant contributions in different fields: logic, topology, economics, operation research, quantum mechanics, probability, weather prediction and computer science. He also had a number of important official positions in the US (see for example [42]).

in [48].

Gödel's incompleteness theorem had a devastating effect on Hilbert, see pg. 287 in [23]. However, some other mathematicians were motivated by the Entscheidungsproblem, such Max Newman (1897-1984), a topologist that attended Hilbert's lecture in Bologna. In 1935 he taught a course at Cambridge University on the foundations of mathematics, where he finished by explaining Gödel's incompleteness theorem and pointing to the students that the Entscheidungsproblem was still open, in the sense that an undecidable statement had still to be found. One of the undergraduates attending the course was Alan Turing. The following year he wrote a paper showing such a statement.

4.2 Solution to the Entscheidungsproblem

In 1936 Turing had ready the draft of his paper "On Computable Numbers, with an Application to the Entscheidungsproblem" [70]. The contribution of the paper is the proposal of a formal model of computation, the *a-machine*, today known as the *Turing machine*[7], and it was proved that any function that could be *effectively computed* (in the sense of Hilbert) by a human computer could also be *mechanically calculated* in a finite number of steps by a mechanical procedure, namely the Turing machine. The paper goes on using this formalism to define the set of *computable numbers*, those reals for which their ith decimal can be computed in a finite number of steps. The number of computable numbers is countable but the number of reals is uncountable, so most of the reals are not computable. Some well-known computable numbers are $1/7, \pi, e, \ldots$.

In his model, Turing introduced two essential assumptions: the discretization of time and the discretization of the state of mind of a human computer. He was able to simulate complicate behavior of the mind in a human computer by a limited number of states. As Turing writes in [70]: "The idea behind digital computers may be explained by saying that these machines are indeed intended to carry out any operations which can be done by a human computer. The human computer is supposed to be following fixed rules, he has no authority to deviate from them in any detail. We may suppose that those rules are supplied in a book, which is altered whenever he is put on to a new job. He has also an unlimited supply of paper on which he does his calculations. He may also do his multiplications and additions on a desk machine, but this is not important."

[7] In 1937 Church wrote a review of Turing's paper in the *Journal of Symbolic Logic* and coined the term Turing machine.

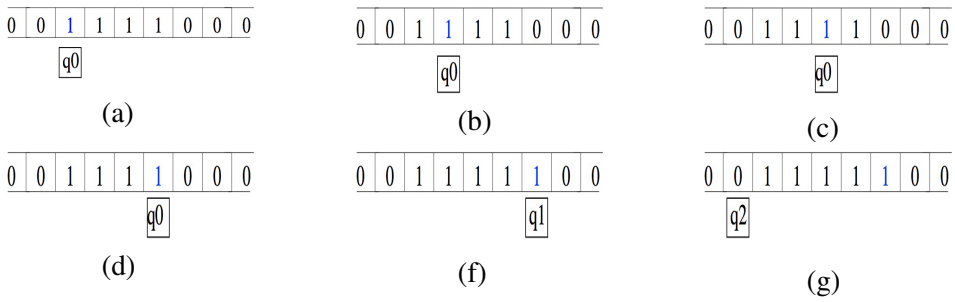

Figure 1: Snapshots of the example of TM computation in the text. (a) Initial position, (b), (c) and (d) scanning the input until the end of 1's, (e) adding a new 1 to the end, (f) final position after going back to 0's at the left.

A decade later, this analysis of a virtual machine based on mental rule operations gave rise to the field of Artificial Intelligence.

A *Turing machine* (TM) consists of a infinite tape divided in squares, a finite input on a finite alphabet and a write-read head that can be in a finite number of states, representing Turing's discretization of the human computer's brain. At each step, the head reads the symbol in the square under it and, depending on the symbol and the current state and using a "stored-in program", erases, writes, moves one square right or left and changes (or not) state.

The following example presents a TM to solve the following problem: "Given $n \in \mathbb{N}$ in unary (as a row of consecutive $(n+1)$ 1's), with 0's to the left and right of the block of 1's, write integer $n+1$ and return to the initial position". The machine has an alphabet $\{0, 1\}$ and three states $\{q_0, q_1, q_2\}$. At the beginning the head is in state q_0 scanning the leftmost 1, the "stored-in program" is the following list of actions: (1) when in state q_0 and scanning a 1 move right, (2) when in state q_0 and scanning a 0, change the 0 to 1, change to state q_1 and move left, (3) when in state q_1 and scanning a 1 move left, (4) when in state q_1 and scanning a 0, change state to q_2, move right and halt. At the end, there would be $(n+2)$ 1's, with the head at the initial position. Figure 1 shows a sequence of pictures of the head and the tape during the execution of the "stored-in program".

Notice that a single TM is finite, so we can encode it as an integer, therefore the number of TM's is infinite but enumerable. As Turing knew well the proof of Gödel's incompleteness theorem, it was natural for him to

encode his machines as integers.

An outstanding achievement of Turing was the concept of *universal Turing machine* (UTM). For example, the chapter devoted to Turing in [18] has the very illustrative title "Turing Conceives the All-Purpose Computer". Given as input the codified description simulate TM $\mathcal{M}$ (the instruction table) and the input to $\mathcal{M}$, the UTM is able to perform the computation of that machine and write the result. So the UTM can simulate any other TM.

The idea of universality from Leibniz and Babbage was finally realized in the *Universal Turing Machine*. Today's computers, from big supercomputers to laptops, are just variations of the universal machine concept.

Another contribution of Turing to computability was to give an example of undecidable problem, the *halting problem*. In today's terminology: "Given a program and an input to it, decide whether the computation will halt on the given input". Turing stated the halting problem in terms of TM's and language recognition. For a nice informal description of the proof that the halting problem is undecidable, see Chapter 7 in [18].

At the time Turing was writing his paper, he was not aware that other mathematicians were working on the same topic. Alonzo Church (1903 –1995) had been for some time trying to formalize the notion of effective computation as well as to provide an example of undecidable problem. Together with his student S. Kleene, he was developing λ-*calculus*, a formalism to characterize the class of efficiently computable functions. In 1936 he published his formal system [10], where he also proved that the problem of deciding a normal form in the λ-calculus was undecidable, therefore providing an example of undecidability of the Entscheidungsproblem. When Turing was finishing his paper, the issue of the *Journal of Symbolic Logic* with Church's paper arrived at Cambridge, and Turing hastily added an appendix to his draft, sketching the equivalence between the concepts of decidability using TM's and λ-calculus. In 1937, Turing would produce a full paper on the equivalence between the two models [71]. He went to the Institute for Advanced Study at Princeton to do a PhD under Church.

It is interesting to notice that Gödel did not believe that λ-calculus provided the full characterization for decidability and, as reported in [61], when Gödel knew Turing's model he wrote: "That this is really the correct definition of mechanical computability was established beyond any doubt by Turing". For enlightening accounts of the historical debate about the two formalisms see [62, 61]. Today, and the equivalence between λ-calculus and the Turing machine as characterizations of decidability is known as the *Church-Turing thesis*. However, the fact that Turing characterizes human

computability in machine terms has an enormous appeal. This doesn't deny the tremendous importance that Church's λ-calculus had for the development of functional programming languages, like Haskell, Scheme and others.

Turing's PhD thesis extended Gödel's incompleteness theorem by proving that adding new axioms to an incomplete formal system, the system remains incomplete [72]. An important contribution of that paper was the *oracle* TM model. In the words of Turing: "Let us suppose we are supplied with some unspecified means of solving number-theoretical problems, a kind of oracle as it were. We shall not go any further into the nature of this oracle apart from saying that it cannot be a machine". An example would help to understand the concept of oracle Turing machine. Consider the *Goldbach's conjecture problem*: "Decide whether it is true that every even integer greater than 2 can be written as the sum of two primes." It is easy to verify the conjecture for small integers, $8 = 3 + 5$ or $1714 = 1361 + 353$. In fact, the conjecture has been empirically verified up to values of $n = 4 \times 10^{18}$ [**?**]. However, since in 1742 Christian Goldbach exposed the problem to Euler, a general solution has remained open. The problem has been a favorite one for many mathematicians and there is a nice novelized presentation of the problem by A. Doxiadis [22].

Assume that we have an oracle H that could decide the halting problem, then we can design a TM that, from $i = 3$ to ∞, at each step increases i by 1 and tests whether $2 * i$ is the sum of two primes, and if it is not, halts. Then one could ask the oracle if the machine would halt, if the answer were yes then the solution to the Goldbach's conjecture would be false.

Turing's oracle machine can be used to compare the relative difficulty of problems. Given problems P_1 and P_2, P_1 is *Turing-reducible*[8] to P_2 ($P_1 \leq_T P_2$) if P_2 can be used to construct in a finite number of steps a program to solve P_1 (for a technical definition of T-reducibility, see for example [56, 5]). Note that if $P_1 \leq_T P_2$ and P_2 is decidable then P_1 is also decidable, and if $P_1 \leq_T P_2$ and P_1 is undecidable then P_2 is also undecidable. Therefore, T-reducibility can be used to enlarge the class of know undecidable problems.

The oracle TM was a key tool to compare degrees of indecidability among problems and led first Post and after Kleene and Mostowski [53, 36] to define the *arithmetic hierarchy*, where problems are classified according to their degree of unsolvability and each class is characterized by an alternation of existential and universal unbounded quantifiers [56].

[8]The name Turing-reducibility is due to Post.

4.3 Post's contribution

Emil Post (1897-1954) was an extraordinary bright mathematician with a severe maniac-depressive mental disease, which affected his life. After enrolling as a graduate student at Columbia University (1918), he attended a seminar on what then was a recently published three-volume work on the foundations of mathematics, namely *Principia Mathematica* by Whitehead and Russell. The seminar had a strong influence on Post, who started his PhD in logics, which he completed in 1921. It is widely accepted that his thesis contained essentially the incompleteness theorem of Gödel, a similar model to the Turing machine, as well as a formulation of what later would be defined as recursive function theory (see [64, 65, 21], the introduction in [16], and problem 7.34 in [46]). According to Hartmanis, "Post was trying to accomplish single handedly, in one blow, what was accomplished by Gödel, Church, Kleene and Turing" [28].

It is worth to remark that Post work did not solve the Entscheidungsproblem nor did he define the universality of his model or its equivalence with the existing models of effective computability. In 1941 Post, who was well aware of the work of Gödel, Chuch and Turing, submitted his work done from the 20's on to the *American Journal of Mathematics*, where the manuscript was rejected for the basic reason that (quoted from [17]): "... twenty years ago your work did not find its due recognition. However, we cannot turn the clock back; in the meantime Gödel, Church and others have done what they have done, and this journal is no place for historical accounts ...". In 1943 a short journal version appeared in [50], and in 1994 the full version was published in a compilation of his work, edited by Martin Davis, who had been his PhD student [17].

In 1946 Post published the undecidability of what today is known as the *Post Correspondence Problem* [52]. He extended Turing's *oracle machine* to define reductions among problems [51]. First Post, and after his death Kleene, used T-reducibility for defining the *arithmetical hierarchy* of classes of equivalent undecidable problems, where each problem in a level of the hierarchy is "a degree more undecidable" than the problems in the previous level.

5 Turing and the digital computer

The purpose of the present work is not a presentation of the early digital computer race, for which there is plenty of literature. For instance, nice sources of information, focused mainly on the computer race in the US,

are [24, 60]. Among other things, these books explain the importance of military in the development of the first digital computers, and how much of this work was kept under secrecy until two decades later.

Our goal in this section is to sketch the role that Turing may have played in the birth of the digital computer. In the standard terminology, a digital computer is said to be *Turing-complete* if it is able to simulate a UTM, i.e., if it is universal. For instance, todays laptops are Turing-complete and the internet can be loosely considered as a network of Turing-complete computers.

As we already mentioned, in the first part of the 20c. there were several sophisticated single-purpose analytical calculators, but an important contribution came in 1941 with the construction of the Atanasoff-Berry digital computer (ABC). John Atanasoff (1903-1995) was a professor of physics that designed the ABC to simplify his mathematical calculations. The ABC used the binary number system with an electronic central processor consisting of electronic tubes to perform basic arithmetic operations. It was not programmable and therefore not Turing-complete.

One of the first program-controlled, Turing-complete digital computers was the Z3 that Conrad Zuse (1910-1995) built in 1941, with little funding from the German government. The Z3 had several features as floating-point arithmetic and the use of a binary system, as Leibniz had put forward four centuries before. The input to the Z3 was introduced through a punched tape. Zuse anticipated, but never implemented, the storage of machine instructions into machine memory. It seems Zuse's ideas were unknown in the US and UK until after WWII, but probably he was aware of Turing's work.

Another early computer in Europe was the Colossus (1943), a digital computer designed at Bletchtley Park to crack one type of German cryptography by a team headed by Max Newman (the old Turing's professor). The Colossus was programmable by punched tape and switches, but it was not Turing-complete, since it was configured to make a series of Boolean operations on its input data. Although Turing was at Bletchley Park at the time, he had little direct involvement in the development of the machine, but his idea of TM may have been a source of inspiration for its design, since Newman knew very well Turing's papers [13]. Due to military secrecy, the Colossus existence was a secret of state until the 1970's[9], although it seems plausible that due to the military cooperation between US and UK, some

[9] In fact, it was dismantled. The current Colossus at Bletchley Park museum is an unfinished reconstruction based on the original blueprints.

high-rank people involved in the war effort at the US could have known the existence of the computer, in particular John von Neumann.

The first Turing-complete digital computer in the US was ENIAC. At the beginning it was aimed to speed up the construction of firing tables, but it was programable through switches and cables, i.e., the program was introduced mechanically by hand into the machine. Its construction started in 1943 under John Mauchly and Presper Eckert, and it was operational in 1946. Another characteristic is that it used the decimal system. In 1973 a federal US judge invalidated the previous patent recognizing that ENIAC was the first digital computer, on the basis that some of the features in ENIAC were present in ABC and that in 1941 Atanasoff had given Eckert a demonstration of ABC's workings [24]. It is also known that von Neumann visited ENIAC in August 1944, at the same time that Mauchly, Eckert and their team were already discussing a new machine.

In June 1945 von Neumann wrote his famous "The First Draft of a Report on the EDVAC" [80], where he described the logical design of a computer using the stored-program concept, which has been known since then as von Neumann's architecture, and which still is the principle of today's computer architecture. In the report, von Neumann did not cite Turing or anybody else. However, it seems plausible that the whole concept of stored-program machine could have been inspired by the Turing machine, which he not only knew very well, but also recommended to engineers working on the development of digital computers. Recall that in the universal Turing machine, the programs controlling it are represented as data, and manipulated by the machine in the same way as the input, which is the basis of the stored-program machine. Moreover, many of the ideas in the report seem to come from the discussions between Mauchly, Eckert and other engineers during the brainstorming meetings to devise a successor for ENIAC [25].

After the war, in 1945, when Turing moved from Bletchley Park to the *National Physical Laboratory* (NPL), his bosses asked him to write a document on the feasibility of building a digital all-purpose computer. The following year he presented to the executive committee of NPL his detailed report: "Proposal for Development in the Mathematics Division of an Automatic Computing Engine" [75], also know as the ACE-report, a very detailed explanation of how to construct an electronic stored-program computer. It introduced several new features, for instance the *Abbreviated Computer Instructions*, a formal programming language for the ACE. One of the bright ideas of Turing to speed up computation when storing machine instructions was the use of direct access to memory registers. Turing also proposed that ACE should have separate registers for performing the basic operations: ad-

dition, multiplication, logical operations, etc. These ideas would have made ACE the fastest computer of its time. In the report, the possibility of programming and using the machine from distance via telephone's lines was also suggested. Despite NPL being an English governmental institution devoted basically to military applications, the members of the executive committee were not aware of the existence of Colossus, and for security reasons Turing could not say that Colossus existed, so the project was downgraded to building a toy ACE machine [13, 9]. Finally, the full machine was completed in 1958, when computer architecture had improved quite a bit and the ACE was already obsolete. For a vivid recount of the events, see [84].

Therefore, it seems clear that the TM influenced von Neumann's report and it is known that von Neumann's report influenced Turing's ACE report. But the importance of Mauchly, Eckert and other engineers that contributed to the development of the early computers should not be downplayed.

Although mathematicians had an important role in the computer's race, for some time most of them (but not all, and in particular, not Turing or von Neumann) tended to consider digital computers as mere tools to crunch data. For instance, according to Hodges' book, at the beginning of the 1950's a grad student at University of Manchester, after reading some of Turing's and Church's papers, asked Max Newman for advise on how to do a PhD using digital computers, and Newman's answer was that "there is nothing to do with computers that merits a PhD" (pg. xv in [32]).

Another important programming contribution by Turing came out while he was at Manchester university as head of the computer laboratory, where he wrote what is considered to be the first document on verification of computer programs. The document, together with annotations, was reprinted in [47].

6 Hypercomputation

Hypercomputation deals with the design of computer models that go beyond the decidability paradigm. There are different kinds of models: the ones that defy physical laws usually the computer is left on earth, while the operator makes a trip in a rocket at a speed close to the one of light, so the time is altered. For a rebuttal of these models see [2, 3].

A second approach relies on machines with special properties, like Zenon machines, fuzzy machines and neural networks that can deal with real non-computable numbers (not just the computable ones defined by Turing). Most of these constitute nice theoretical models of computation, but as

regards to somehow "solve" undecidable problems in the near future, Davis said it beautifully in [19]: "The evidence provided by hypercomputation amounts to the trivial remark that given an oracle that makes uncomputable information available, it would become possible to compute uncomputable functions."

7 Conclusions

Alan Turing was a singular scientist that left a deep imprint in the history of computing and whose premature death probably deprived us of other important contributions. His work is undoubtedly an important link in the chain of research leading to the development of the digital computer. In 2012, the year of the centenary of his birth [1], he has been chosen as a banner to honor also the many other scientists who contributed in various degrees to that chain: from Leibniz, with his dream of a universal method of reasoning, to Babbage and Lovelace, Hilbert and Gödel, and especially Turing's contemporary, namely von Neumann, Newman, Eckert, Mauchly, Kleene, Church and Post, as well as many other researchers without whom the world today would not be as digital as it is. In this paper we have tried to acknowledge the contributions of these top researchers, by articulating them in a historical account of the birth and early evolution of the computer science field.

References

[1] 2012 the Alan Turing year. http://www.mathcomp.leeds.ac.uk/turing2012/.

[2] S. Aaronson. NP-complete problems and physical reality. Technical Report arXiv:quant-ph/0502072v2, 2005.

[3] S. Aaronson and J. Watrous. Closed timelike curves make quantum and classical computing equivalent. Technical Report arXiv:quant-ph/0808.2669, 2009.

[4] C. Babbage. *Passages from the Life of a Philosopher*. Cambridge Library Collection, 2011.

[5] J. L. Balcazar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Springer-Verlag, 2 edition, 1990.

[6] A. Bonner. *The art and logic of Ramon Llull*. Brill Academic Publishing, 2007.

[7] A. Booker. Turing and the Riemman hypothesis. *Notices of the American Math. Society*, 53(10):1208–1211, 2006.

[8] A. G. Bromley. Charles Babbage's analytical engine, 1838. *Annals of the History of Computing*, 4(3):196–217, 1982.

[9] B. Carpenter and R. Doran. The other Turing machine. *The Computer Journal*, 20(3):296–279, 1977.

[10] A. Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1(1):40–41, 1936.

[11] P. Cockshott, L. Mackenzie, and G. Michaelson. *Computation and its Limits*. Oxford University Press, 2012.

[12] B. Cohen. Babbage and Aiken. *Annals of the History of Computing*, 33(1):22–37, 2011.

[13] J. Copeland. *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*. Oxford University Press, 2006.

[14] G.-G. Coriolis. Note sur un moyen de tracer des courbes donn?s per des equations differentielles. *Journal de Mathematiques Pures et Appliqu?s*, 1:5–9, 1836.

[15] F. Cucker. The legacy of Turing in numerical analysis. In M. Bielikova, G. Friedrich, G. Gottlob, S. Katzenbeisser, and G. Turan, editors, *Proc. SOFSEM*, volume 7147 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2012.

[16] M. Davis. *The Undecidable*. Raven Press, 1965.

[17] M. Davis. *Solvability, provability, definability: The collected works of Emil L. Post*. Birkhauser, 1994.

[18] M. Davis. *The universal computer: the road from Leibniz to Turing*. Norton, 2000.

[19] M. Davis. Why there is no such discipline as hypercomputation. *Applied Mathematics and Computation*, 178:4–7, 2006.

[20] J. W. Dawson. *Logical Dilemmas: The Life and Work of Kurt Godel*. AK Peters, 1996.

[21] L. de Mol. Closing the circle: An analysis of Emil Post's early work. *The Bulletin of Symbolic Logic*, 12(2):267–289, 2006.

[22] A. Doxiadis. *Uncle Petros and Goldbach's Conjecture*. Bloomsbury, 2001.

[23] A. Doxiadis, C. Papadimitriou, A. Papadatos, and A. di Donna. *LOGICOMIX: an epic search for truth*. Bloomsbury, 2009.

[24] G. Dyson. *Turing's Cathedral*. Random House, 2012.

[25] M. D. Godfrey and D. F. Hendry. The computer as von Neumann planned it. *Annals of the History of Computing*, 15:11–21, 1993.

[26] D. Grier. *When computers were human*. Princenton University Press, 2005.

[27] K. Gödel. Uber formal unentscheidbare satze der principia mathematica und verwandter systeme. *Monatshefte fur Mathematik und Physik*, 38:173–198, 1931.

[28] J. Harmanis. Gödel, von Neumann and the P=?NP problem. *Bulletin of the European Association for Theoretical Computer Science*, 38:101–107, 1989.

[29] J. V. Heijenoort, editor. *From Frege to Gödel: A Source Book in Mathematical Logic*. Harvard University, 1967.

[30] D. Hilbert. Mathematical problems. *Bulletin of the American Mathematical Society*, 8(10):437–479, 1970.

[31] D. Hilbert and W. Ackermann. *Grundzüge der Theoretischen Logik*. Julius Springer, 1928.

[32] A. Hodges. *Alan Turing: The Enigma*. Random House, 2nd. edition, 1992.

[33] A. Hodges. Book Review: The essential Turing. *Notices of the ACM*, 53(10):1190–1199, 2006.

[34] D. Hofstadter. *G?el, Escher, Bach: An Eternal Golden Braid*. Basic Books, 1979.

[35] P. Kidwell. Collected works of A. M. Turing — morphogenesis. *Annals of the History of Computing*, 18(4):69–69, 1996.

[36] S. C. Kleene and E. L. Post. The upper semi-lattice of degrees of recursive unsolvability. *Annals of Mathematics*, 59:379–407, 1954.

[37] D. E. Knuth. Ancient babylonian algorithms. *Commun. ACM*, 15(7):671–677, 1972.

[38] D. E. Knuth. Al-khorezmi. In A. P. Ershov and D. Knuth, editors, *Algorithms in Modern Mathematics and Computer Science*, volume 122 of *Lecture Notes in Computer Science*, pages 82–99. Springer-Verlag, 1980.

[39] R. Lewin. *Ultra Goes to War: The Secret Story*. Hutchinson, London, 1978.

[40] M. Lindgren. *Glory and Failure: The Difference Engines of Johann Muller, Charles Babbage and Georg and Edvard Scheutz*. MIT Press, 1990.

[41] P. Ludgate. On a proposed analytical machine. *Scientific Proceedings of Royal Dublin Society*, 12(9):77–91, 1909.

[42] N. Macrae. *The Scientific Genius Who Pioneered the Modern Computer, Game Theory, Nuclear Deterrence, and Much More*. American Mathematical Society, 1999.

[43] B. Meltzer. *On Formally Undecidable Propositions of Principia Mathematica and Related Systems. Translation of the German original by Kurt Godel*. Dover, 1992.

[44] L. Menabrea. The analytical engine invented by Charles Babbage. Technical report, http://www.fourmilab.ch/babbage/sketch.html, 1842.

[45] H. Midonick. *The Treasury of Mathematics: 1*. Penguin, 1968.

[46] C. Moore and S. Mertens. *The Nature of Computation*. Oxford University Press, 2011.

[47] F. L. Morris and C. B. Jones. An early program proof by Alan Turing. *Annals of the History of Computers*, 6(2):139–143, 1984.

[48] E. Nagel and J. R. Newman. *G?el's Proof*. Routledge, 1989.

[49] T.Oliveira. Goldbach's conjecture verification. http://www.ieeta.pt/ tos/goldbach.html

[50] E. L. Post. Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics*, 65:197–215, 1943.

[51] E. L. Post. Recursively enumerable sets of positive integers and their decision problem. *Bulletin of the American Mathematical Society*, 50:284–316, 1944.

[52] E. L. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52:264–268, 1946.

[53] E. L. Post. Degrees of recursive unsolvability: preliminary report. *Bulletin of the American Mathematical Society*, 54:641–642, 1948.

[54] B. Randell. From analytical engine to electronic digital computer: The contributions of ludgate, torres, and bush. *Annals of the History of Computing*, 4(4):327–341, 1982.

[55] C. Reid. *Hilbert*. Springer, 1996.

[56] H. Rogers. *Theory of Recursive Functions and Effective Computability*. MIT Press, 1987.

[57] T. Sales. Llull as computer scientist. In A. Fidora and C. Sierra, editors, *Ramon Llull: From the Ars Magna to Artificial Intelligence*, pages 25–37. IIA-CSIC, 2011.

[58] P. T. Saunders. Alan Turing and biology. *Annals of the History of Computing*, 15(3):33–36, 1993.

[59] S. Singh. *The Code Book*. Anchor Books, 2000.

[60] J. Smiley. *The man who invented the computer: the biography of John Atanasoff, digital pioneer*. 2010.

[61] R. I. Soare. Turing and Michelangelo: The art of classical computability. In B. Cooper and J. van Leewen, editors, *Alan Turing- His work and impact*, pages 182–199. Elsevier, 2012.

[62] R. I. Soare. Turing computability and information content. *Philosophical Transactions of the Royal Society*, A370:3277–3304, 2012.

[63] D. Spinellis. The Antikythera mechanism: A computer science perspective. *Computer*, 41(5):22–27, 2008.

[64] J. Stillwell. Emil Post and his anticipation of Godel and Turing. *Mathematics Magazine*, 77(1):3–14, 2004.

[65] J. Stillwell. *Roads to Infinite: The Mathematics of Truth and Proof*. AK Peters, 2010.

[66] D. Swade. *The cogwheel brain*. Abacus, 2000.

[67] D. Swade. *The Difference Engine: Charles Babbage and the Quest to Build the First Computer*. Penguin, 2002.

[68] F. Thomas. A short account of Leonardo Torres' endless spindle. *Mechanism and Machine Theory*, 43(8):1055–1063, 2008.

[69] L. Torres-Quevedo. Ensayos sobre automatica-su definicion. extension teorica de sus aplicaciones. *Revista de la Real Academia de Ciencias Exactas, Fisicas y Naturales*, 12:391–418, 1913.

[70] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society-2*, 42:230–265, 1936.

[71] A. M. Turing. Computability and lambda-definability. *Journal of Symbolic Logic*, 2(4):153–163, 1937.

[72] A. M. Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society-2*, 45:161–228, 1939.

[73] A. M. Turing. The applications of probability to cryptography. Report, GCHQ, Cheltenham, UK, 1941.

[74] A. M. Turing. On statistics of repetitions. Report, GCHQ, Cheltenham, UK, 1941.

[75] A. M. Turing. Proposal for development in the Mathematics Division of an Automatic Computing Engine (ACE). Report E.882, Executive Committee, inst-NPL, 1945.

[76] A. M. Turing. Rounding-off errors in matrix processes. *The Quarterly Journal of Mechanics and Applied Mathematics*, 1 (part 3):287–308, Sept. 1948.

[77] A. M. Turing. Computing machinery and intelligence. *MIND*, 59(236):433–460, 1950.

[78] A. M. Turing. The chemical basis of morphogenesis. *Philosophical Trans. of the Real Society of London (Series B)*, B 237(641):37–72, Aug. 1952.

[79] A. M. Turing. Some calculations of the Riemann zeta-function. *Proceedings of the London Mathematical Society*, 3(3):99–117, 1953.

[80] J. von Neumann. The first draft of a report on the EDVAC. http://qss.stanford.edu/~godfrey/vonNeumann/vnedvac.pdf, 1945.

[81] J. Walker. Babbage machines. http://www.fourmilab.ch/babbage.

[82] A. Whitehead and B. Russell. *Principia mathematica; 2nd ed.* Cambridge University Press, 1927.

[83] M. V. Wilkes. Babbage as a computer pioneer. *Historia Mathematica*, 4(4):415–440, 1977.

[84] J. H. Wilkinson. Turing's work at the National Physical Laboratory and the construction of Pilot ACE, DEUCE, and ACE. In N. Metropolis, J. Howlett, and G.-C. Rota, editors, *A History of Computing in the Twentieth Century: A Collection of Essays*, pages 101–114. Academic Press, 1980.

[85] S. L. Zabell. Alan Turing and the central limit theorem. *American Mathematical Monthly*, 102(6):483–494, 1995.

[86] H. Zemanek. Al-khorezmi. In A. P. Ershov and D.E.Knuth, editors, *Algorithms in Modern Mathematics and Computer Science*, volume 122 of *Lecture Notes in Computer Science*, pages 1–81. Springer-Verlag, 1980.