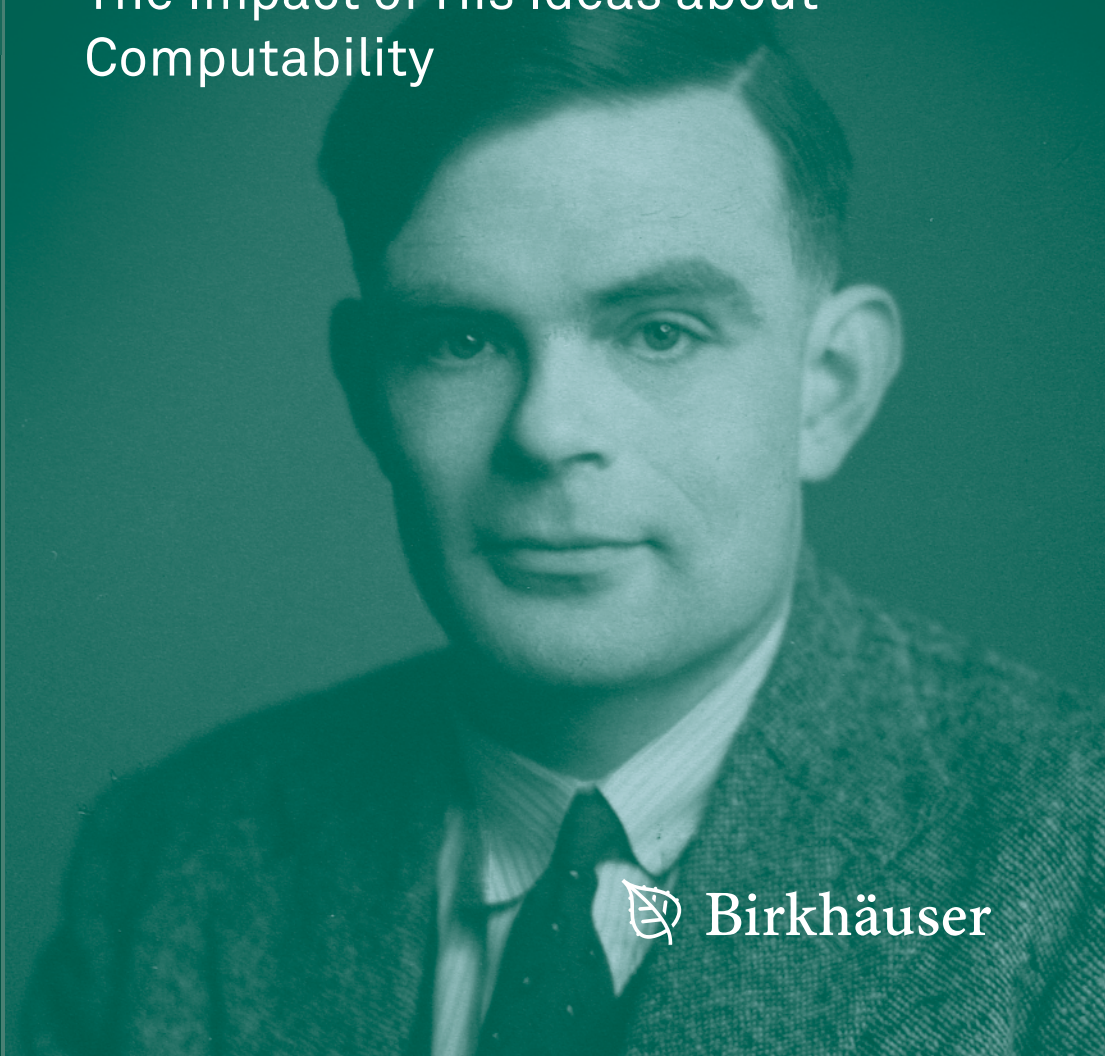
Giovanni Sommaruga
Thomas Strahm
Editors

Turing's Revolution

The Impact of His Ideas about Computability



Birkhäuser

Turing's Revolution

Birkhäuser

Giovanni Sommaruga • Thomas Strahm
Editors

Turing's Revolution

The Impact of His Ideas about Computability

Birkhäuser

Editors
Giovanni Sommaruga
Department of Humanities, Social and Political Sciences
ETH Zurich
Zurich, Switzerland

Thomas Strahm
Institute of Computer Science and Applied Mathematics
University of Bern
Bern, Switzerland

ISBN 978-3-319-22155-7 ISBN 978-3-319-22156-4 (eBook)
DOI 10.1007/978-3-319-22156-4

Library of Congress Control Number: 2015958070

Mathematics Subject Classification (2010): 03-XX, 03Axx, 03Dxx, 01Axx

Springer Cham Heidelberg New York Dordrecht London

Cover illustration: Image number: RS.7619, Credit: © Godrey Argent Studio
Cover design: deblik, Berlin

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.birkhauser-science.com)

Preface

June 23, 2012 was Alan Turing's 100th birthday. In the months preceding and following this date, there were widespread events commemorating his life and work. Those of us who had been writing about his work received more invitations to participate than we could accommodate. Speakers at one of the many Turing conferences were often invited to contribute to an associated volume of essays. This book developed from such a meeting, sponsored by the Swiss Logic Society, in Zurich in October 2012. The table of contents hints at the breadth of developments arising from Turing's insights. However, there was little public understanding of this at the time of Turing's tragic early death in 1954.

Thinking back on my own 65 years of involvement with Turing's contributions, I see my own dawning appreciation of the full significance of work I had been viewing in a narrow technical way, as part of a gradual shift in the understanding of the crucial role of his contributions by the general educated public. In 1951, learning to program the new ORDVAC computer shortly after I had taught a course on computability theory based on Turing machines, it became clear to me that the craft of making computer programs was the same as that of constructing Turing machines. But I was still far from realizing how deep the connection is. In the preface to my 1958 *Computability & Unsolvability*, I wrote:

> The existence of universal Turing machines confirms the belief ... that it is possible to construct a single "all purpose" digital computer on which can be programmed (subject of course to limitations of time and memory capacity) any problem that could be programmed for any conceivable deterministic digital computer.... I was very pleased when [it was] suggested the book be included in [the] Information Processing and Computers Series.

My next publication that commented explicitly on Turing's work and its implications was for a collection of essays on various aspects of current mathematical work that appeared in 1978. My essay [3] began:

> ... during the Second World War ... the British were able to systematically decipher [the German] secret codes. These codes were based on a special machine, the "Enigma" ... Alan Turing had designed a special machine for ... decoding messages enciphered using the Enigma.

> ... around 1936 [Turing gave] a cogent and complete logical analysis of the notion of "computation." ... [This] led to the conclusion that it should be possible to construct "universal" computers which could be programmed to carry out any possible computation.

The absolute secrecy surrounding information about the remarkable British effort to break the German codes had evidently been at least partially lifted. It began to be widely realized that Turing had made an important contribution to the war effort. For myself, I had come to understand that the relationship between Turing's mathematical construction that he called a "universal machine" and the actual computers being built in the postwar years was much more than a suggestive analogy. Rather it was the purely mathematical abstraction that suggested that, given appropriate underlying technology, an "all-purpose" computing device could be built.

The pieces began to come together when the paper [1], published in 1977, explained that Turing had been involved in a serious effort to build a working general purpose stored-program computer, his Automatic Computing Engine (ACE). Andrew Hodges's wonderful biography (1983) [5] filled in some of the gaps. Finally Turing's remarkably prescient view of the future role of computers became clear when, in [2], Turing's actual design for the ACE as well as the text of an address he delivered to the London Mathematical Society on the future role of digital computers were published. It was in this new heady atmosphere that I wrote my essay [4] in which I insisted that it was Turing's abstract universal machine, which itself had developed in the context of decades of theoretical research in mathematical logic, that had provided the underlying model for the all-purpose stored program digital computer. This was very much against the views then dominant in publications about the history of computers, at least in the USA. Meanwhile, noting that 1986 marked the 50th anniversary of Turing's universal machine, Rolf Herken organized a collection of essays in commemoration. There were 28 essays by mathematicians, physicists, computer scientists, and philosophers, reflecting the breadth and significance of Turing's work.

June 28, 2004, 5 days after Turing's 90th birthday, was Turing Day in Lausanne. Turing Day was celebrated with a conference at the Swiss Federal Institute of Technology organized by Christof Teuscher, at the time still a graduate student there. There were nine speakers at this one-day conference with over 200 attendees. Teuscher also edited the follow-up volume of essays [7] to which there were 25 contributors from remarkably diverse fields of interest. Of note were two essays on generally neglected aspects of Turing's work. One, written by Teuscher himself, was about Turing's work on neural nets written in 1948. The other recalled Turing's work during the final month's of his life on mathematical biology, particularly on morphogenesis: Turing had provided a model showing how patterns such as the markings on a cow's hide could be produced by the interactions of two chemicals. A more bizarre topic, discussed in a number of the essays, which had developed at the time under the name *hypercomputation*, sought ways to compute the uncomputable, in effect proposing to carry out infinitely many actions in a finite time. This was particularly ironic because computer scientists were finding that mere

Turing computability was insufficient for practical computation and were seeking to classify algorithms according to criteria for feasibility.

As Turing's centenary approached, there was a crescendo of events bringing Turing's importance to the attention of the public. An apology by the British government for his barbaric mistreatment along with a commemorative postage stamp was a belated attempt to rectify the unrectifiable. I myself spoke about Turing at various places in the USA and Britain and also in Mexico, Peru, Belgium, Switzerland, and Italy, crossing and recrossing the Atlantic Ocean several times in a few months' time. A high point was attending a banquet at King's College, Cambridge (Turing's college), with members of Turing's family present, celebrating his actual 100th birthday. I'm very fond of Zurich and was delighted to speak at the conference there from which this volume derives. The perhaps inevitable consequence of Alan Turing's developing fame was Hollywood providing in *The Imitation Game* its utterly distorted version of his life and work.

Considering the authors whose contributions the editors have managed to gather for this volume, I feel honored to be included among them. When it turned out that my good friend Wilfried Sieg and I were writing on very similar subjects, we agreed to join forces, providing for me a very enjoyable experience. Among the varied contributions, I particularly enjoyed reading the essay by Jack Copeland and Giovanni Sommaruga on Zuse's early work on computers. I had been properly chastised for omitting mention of Zuse in my *The Universal Computer*, and I did add a paragraph about him in the updated version of the book for Turing's centenary. However, I had never really studied his work and I have learned a good deal about it from this excellent essay. The writings of Sol Feferman can always be depended on to show deep and careful thought. I very much enjoyed hearing Stewart Shapiro's talk at the conference in Zurich, and I look forward to reading his essay. Barry Cooper has, of course, been a driving force in the 2012 celebrations of Alan Turing and in the "Computability in Europe" movement. Finally, I want to thank Giovanni Sommaruga and Thomas Strahm for bringing together these authors and editing this appealing volume.

Berkeley, CA, USA Martin Davis
February 2015

References

1. B.E. Carpenter, R.W. Doran, The other Turing machine. Comput. J. **20**, 269–279 (1977)
2. B.E. Carpenter, R.W. Doran (eds.), *A.M. Turing's ACE Report of 1946 and Other Papers* (MIT Press, New York, 1986)
3. M. Davis, What is a computation, in *Mathematics Today: Twelve Informal Essays,* ed. by L.A. Steen (Springer, New York, 1978), pp. 241–267
4. M. Davis, Mathematical logic and the origin of modern computers, in *Studies in the History of Mathematics* (Mathematical Association of America, Washington D.C., 1987), pp. 137–165. Reprinted in [6], pp. 149–174

5. A. Hodges, *Alan Turing, The Enigma* (Simon and Schuster, New York, 1983)
6. R. Herken (ed.), *The Universal Turing Machine - A Half-Century Survey* (Verlag Kemmerer & Unverzagt/Oxford University Press, Hamburg, Berlin/Oxford, 1988)
7. C. Teuscher (ed.), *Alan Turing: Life and Legacy of a Great Thinker* (Springer, Berlin, 2004)

Introduction

During the beginning and about half of Turing's short life as a scientist and philosopher, the notions of computation and computability dazzled the mind of a lot of the most brilliant logicians and mathematicians. Some of the most prominent figures of this group of people are Alonzo Church, Kurt Gödel, Jacques Herbrand, Stephen Kleene, and Emil Post—and of course Alan Turing. The second quarter of the twentieth century was such a melting pot of ideas, conceptions, theses, and theories that it is worthwhile to come back to it and to dive into it over and over again. That's what this volume's first part is about.

What is the point of looking back, of turning back to the past? What is the good of looking at the history, e.g., of the notion of computability? The look backwards serves to better understand the present. One gets to understand the origin, the core ideas, or notions at the beginning of a whole development. The systematical value of the historical perspective may also lie in the insight that there is more to the past than being the origin of the present, that there is a potential there for alternative developments, that things could have developed differently, and even that things could still develop differently. The past and especially a look at the past may thereby become a source of inspiration for new developments. Looking at the past may contribute to planning the future, if one starts to wonder: Where do we come from? Where are we going to? And where do we want to go?

In the second half of the last century and up to this very day, these ideas, conceptions, etc. and in particular Turing's gave rise to a plethora of new logical and mathematical theories, fields of research, some of them extensions or variations, and others novel applications of Turing's conceptions and theories. This volume's second part aims at presenting a survey of a considerable part of subsequent and contemporary logical and mathematical research influenced and sparked off, directly or indirectly, by Turing's logical mathematical ideas, ideas concerning computability and computation.

These generalizations can take on different shapes and go in different directions. It is possible to generalize concepts, theses, or theories of Turing's (e.g., generalizing his thesis to other structures, or his machine concept to also account for the infinite). But it is equally possible to take Turing's fulcrum and by varying it

think of new alternatives to it. Furthermore, seeds of notions or conceptions can be unfolded into full-blown theories (e.g., of Turing degrees). Or one can take a problem of Turing's and search for or discover solutions to it, new and different from Turing's own solution (e.g., transfinite rec. progressions). And what is the meaning of those generalizations in a broad sense? They all can be understood as explorations of Turing's own rich conceptual space (even still roughly restricted to the topic of computability). They all mean to further research into its wealth and its many dimensions, to analyze and transgress it, to widen and enrich it. And they all hint at the "fact" that there is yet more to be found in this mine of technical ideas.

Thus, Part I goes back to the roots of Turing's own as well as to the ideas of some of his contemporaries. Part II is dedicated to the utterly prolific outgrow of these ideas. Bringing Parts I and II together raises some questions such as: How was it possible that these original ideas, conceptions, and theories lead to such an explosion of new and highly innovative approaches in computability theory and theoretical computer science? How is this phenomenon to be understood? And what does it mean? Is there perhaps even more to this past which might be fruitful and of great interest in and for the future? A reflection on computability theory in general and Turing's in particular is what Part III of this volume is all about.

Turing's discoveries, important mathematical logical results and theses, etc. have also set off more penetrating, more daring, or more speculative questions reaching to the outskirts or to the core of mathematics. They raise questions concerning the computable and the incomputable, and concerning the meaning of computability and incomputability w.r.t. the real world. And the philosophical entanglements of Turing's thesis stir up questions such as: What is a proof? What is a theorem and what is a thesis? And different Turing ways lead to the same question, namely: What is mathematics? And what are its visions and its dreams? This needn't be idle speculation. If well and carefully done, it may in turn spark off new mathematical ideas and/or new mathematical research insights. But even if it doesn't, there can be plenty of meaning to it.

The spirit of the first part might be characterized as: There's a future to the past. The second part's spirit may be captured by: There's a future and a past to the present. And the third part's spirit might be put in the formula: Even if there isn't a path from the present to the future, there needn't by no means be nothing.

This volume makes no claim to completeness of coverage of the whole range of mathematical and logical research initiated or triggered off by Turing's respective ideas: Nothing is said or written about Turing and computational complexity theory, no mention is made of Turing computation and probability theory either, to list just a couple of examples. Even less so do Turing's ideas on number theory, cryptology, artificial intelligence, or mathematical biology and the offsprings of those ideas play any role in this volume.

Let's now turn to what actually can be found in this volume:

In 1936, two models of computation were proposed in England and in the USA, resp., which are virtually identical. Davis and Sieg claim that this is not a mere coincidence, but a natural consequence of the way in which Turing and

Post conceived of steps in mechanical procedures on finite strings. Davis and Sieg also claim that the unity of their approaches, Post's mathematical work on normal forms of production systems on the one side, and Turing's philosophical analysis of mechanical procedures on the other, is of deep significance for the theory of computability. Moreover, Davis and Sieg point out that the investigation by mathematical logicians like Hilbert, Bernays, Herbrand, Gödel, and others of mechanical procedures on syntactic configurations were following a line of research very different from the one of Post and Turing. While the former were concentrating on paradigmatic recursive procedures for calculating the values of number-theoretic functions, the latter were struck by the fundamental and powerful character of string manipulations. In what Davis and Sieg call "Turing's Central Thesis," Turing correlates the informal notion of "symbolic configuration" with the precise mathematical one of "finite string of symbols." This thesis has methodologically the problematic status of a statement somewhere between a definition and a theorem. Davis and Sieg finally raise the question whether there is a way of avoiding appeal to such a "Central Thesis," and answer it tentatively positively by referring to Sieg's representation theorem for computable discrete dynamical systems.

Thomas' paper focuses on a specific and also very prominent aspect of Turing's heritage, namely on his analysis of a machine or algorithm, that is "a process of symbolic computation fixed by an unambiguous and finite description."[1] Thomas outlines the history of algorithms starting with what he calls its prehistory and its most important figures, Al-Khwarizmi and Leibniz. The history proper may already start with Boole and Frege, but it decidedly starts with Hilbert and his Entscheidungsproblem, i.e. the problem whether there exists an algorithm for deciding the validity or satisfiability resp. of a given logical formula. Turing in 1936 solved this problem in the negative, and he did this by a radical reduction of the notion of algorithm to some very elementary steps of symbolic computation (later called a Turing machine). According to Thomas, Turing's work and also the one of his contemporaries Church, Kleene, Post, and others ended a centuries-old struggle for an understanding of the notion of algorithm and its range of applicability called computability. This success was made possible by merging two traditions in symbolic computation, namely arithmetic and logic. But for Thomas, 1936 not only marks a point of final achievement, but also a point of departure for the new and rapidly growing discipline of computer science, the starting point of what might be conceived of as posthistory. He claims that the subsequent rise of this new discipline changed the conception of algorithm in two ways: "First, algorithmic methods in computer science transcend the framework of symbolic manipulation inherent in Turing's analysis," and Thomas then offers an overview of such methods. Second, he emphasizes that algorithms are today understood in the context of highly complex software systems governing our life. "The span of orders of magnitude realized in huge data conglomerates and software systems" is so gigantic that diverse

[1]The quotations in this and the following sections are referring to the article which is summarized in the resp. section.

methodologies are required in computer science in order to study the rich landscape of computer systems.

After presenting a biographical sketch of Konrad Zuse, Copeland and Sommaruga "outline the early history of the stored-program concept in the UK and the US," and "compare and contrast Turing's and John von Neumann's contributions to the development of the concept." They go on to argue that, contrary to the recent prominent suggestions, the stored-program concept played a key role in computing's pioneering days, and they provide a logical analysis of the concept, distinguishing four different layers (or "onion skins") comprising the concept. This layered model allows them to classify the contributions made by Turing, von Neumann, Eckert and Mauchly, Clippinger, and others, and especially Zuse. Furthermore, Copeland and Sommaruga discuss whether Zuse developed a universal computer (as he himself claimed) or rather a general-purpose computer. In their concluding remarks, Copeland and Sommaruga "reprise the main events in the stored-program concept's early history." The history they relate begins in 1936, the date of publication of Turing's famous article "On Computable Numbers," and also of Zuse's first patent application for a calculating machine, and runs through to 1948, when the first electronic stored-program computer—the Manchester "Baby"—began working.

Feferman starts with a detailed summary and discussion of versions of the Church-Turing Thesis (CT) on concrete structures given by sets of finite symbolic configurations. He addresses the works of Gandy, Sieg as well as Dershovitz and Gurevich, which all have in common that they "proceed by isolating basic properties of the informal notion of effective calculability or computation in axiomatic form and proving that any function computed according to those axioms is Turing computable." Feferman continues to note that generalizations of CT to abstract structures must be considered as theses for algorithms. He starts by reviewing Friedman's approach for a general theory of computation on first order structures and the work of Tucker and Zucker on "While" schemata over abstract algebras. Special emphasis is put on the structure of the real numbers; in this connection, Feferman also discusses the Blum, Shub, and Smale model of computation. A notable proposal of a generalization of CT to abstract structures is the Tucker–Zucker thesis for algebraic computability. The general notion of algorithm it uses leads to the fundamental question "What is an algorithm?," which has been addressed under this title by Moschovakis and Gurevich in very different ways. Towards the end of his article, Feferman deals with a specific approach to generalized computability, which has its roots in Platek's thesis and uses a form of least fixed point recursion on abstract structures. Feferman concludes with a sensible proposal, the so-called "Recursion thesis," saying that recursion on abstract first order structures (with Booleans {T,F}) belongs to the class he calls Abstract Recursion Procedures, ARP (formerly Abstract Computation Procedures, ACP).

Tucker and Zucker survey their work over the last few decades on generalizing computability theory to various forms of abstract algebras. They start with the fundamental distinction between abstract and concrete computability theory and emphasize their working principle that "any computability theory should be focused equally on the data types and the algorithms." The first fundamental notion is

the one of While computation on standard many sorted algebras, which is a high level imperative programming language applied to many sorted signatures. After a precise syntactical and semantical account of this language, Tucker and Zucker address notions of universality. An interesting question is to consider various possible definitions of semi-computability in the proposed setting. As it turns out, different characterizations, extensionally equivalent in basic computability theory over the natural numbers, are different in the Tucker and Zucker setting. A further emphasis in the paper is laid on data types with continuous operations like the reals, which leads to a general theory of many sorted partial topological algebras. Tucker and Zucker conclude by comparing their models with related abstract models of computation and by proposing various versions of a generalized Church–Turing thesis for algebraic computability.

Welch gives a broad survey of models of infinitary computation, many of which are rooted in infinite time Turing machines (ITTMs). It is the latter model that has sparked a renewed interest in generalized computability in the last decade. After explaining the crucial notion of "computation in the limit," various important properties of ITTMs are reviewed, in particular, their relationship to Kleene's higher type recursion. Furthermore, Welch elaborates on degree theory and the complexity of ITTM computations as well as on a close relationship between ITTMs, Burgess' quasi-inductive definitions, and the revision theory of truth. He then considers variants of the ITTM model that have longer tapes than the standard model. Afterwards Welch turns to transfinite generalizations of register machines as devised by Sheperdson and Sturgis, resulting in infinite time register machines (ITRMs) and ordinal register machines (ORMs); the latter model also has registers for ordinal values. The last models he explains are possible transfinite versions of the Blum–Shub–Smale machine, the so-called IBSSMs. Welch concludes by mentioning the extensional equivalence (on omega strings) of continuous IBSSMs, polynomial time ITTMs, and the safe recursive set functions due to Beckmann, Buss, and Friedman.

Gurevich reviews various semantics-to-syntax analyses of the so-called species of algorithms, i.e., particular classes of algorithms given by semantical constraints. In order to find a syntactic definition of such a species, e.g., a machine model, one often needs a fulcrum, i.e., a particular viewpoint to narrow down the definition of the particular species in order to make a computational analysis possible. Gurevich starts with Turing's fundamental analysis of sequential algorithms performed by idealized human computers. According to Gurevich, Turing's fulcrum was to "ignore what a human computer has in mind and concentrate on what the computer does and what the observable behavior of the computer is." Next, Gurevich turns to the analysis of digital algorithms by Kolmogorov in terms of Kolmogorov–Uspenski machines and identifies its fulcrum, namely that computation is thought of as "a physical process developing in space and time." Then Gurevich discusses Gandy's analysis of computation by discrete, deterministic mechanical devices and identifies its fulcrum in Gandy's Principle I, according to which the representation and working of mechanical devices must be expressible in the framework of hereditarily finite sets. The fourth example discussed is the author's own analysis of the species of sequential algorithms using abstract state machines. Its fulcrum

is: "Every sequential algorithm has its native level of abstraction. On that level, the states can be faithfully represented by first-order structures of fixed vocabulary in such a way that the transitions can be expressed naturally in the language of that fixed vocabulary."

Turing (implicitly) introduced the notion of Turing reducibility in 1939 by making use of oracle Turing machines. This preorder induces an equivalence relation on the continuum, which identifies reals with the same information content and whose equivalence classes are the so-called Turing degrees. Barmpalias and Lewis survey order-theoretic properties of degrees of typical reals, whereby sensible notions of typicality are derived from measure and category. Barmpalias and Lewis present a detailed history of measure and category arguments in the Turing degrees as well as recent results in this area of research. Their main purpose is to provide "an explicit proposal for a systematic analysis of the order theoretically definable properties satisfied by the typical Turing degree." Barmpalias and Lewis identify three very basic questions which remain open, namely: (1) Are the random degrees dense? (2) What is the measure of minimal covers? (3) Which null classes of degrees have null upward closure?

Beklemishev's paper relates to Turing's just mentioned paper entitled "Systems of logic based on ordinals," which is very well known for its highly influential concepts of the oracle Turing machine and of relative computability. The main body of Turing's 1939 paper, however, belongs to a different area of logic, namely proof theory. It deals with transfinite recursive progressions of theories in order to overcome Gödelian incompleteness. Turing obtained a completeness result for Π_2 statements by iterating the local reflection principle, whereas Feferman in 1962 established completeness for arbitrary arithmetic statements by iteration of the uniform reflection principle. Turing's and Feferman's results have the serious drawback that the ordinal logics are not invariant under the choice of ordinal representations and their completeness results depend on artificial such representations. Beklemishev's approach is to sacrifice completeness in favor of natural choices of ordinals. He obtains a proof-theoretic analysis of the most prominent fragments of first order arithmetic by using the so-called smooth progressions of iterated reflection principles.

Juraj Hromkovic's main claim is that to properly understand Alan Turing's contribution to science, one ought to understand what mathematics is and what the role of mathematics in science is. Hromkovic answers these latter questions by comparing mathematics with a new, somewhat artificial language: "one creates a vocabulary, word by word, and uses this vocabulary to study objects, relationships" and whatever is accessible to the language of mathematics at a certain stage of its development. For Leibniz, mathematics offered an instrument for automatizing the intellectual work of humans. One expresses part of reality in the language of mathematics or in a mathematical model, and then one calculates by means of arithmetic. The result of this calculation is again a truth about the investigated part of reality. Leibniz' "dream was to achieve the same for reasoning." He was striving for a formal system of reasoning, a formal system of logic, analogous to arithmetic, the formal system for the calculation with numbers. Hilbert's dream of the perfection

of mathematics involved the idea that every piece of knowledge expressible as a statement in the language of mathematics could be determined to be true or false, and that "for each problem expressible in the language of mathematics, there exists a method for solving it." In particular, there has to be a method which for every true mathematical statement yields a proof of the truth of it. Hilbert's dream was unlike Leibniz' dream restricted to mathematics. New concepts and subsequently new words are introduced into the mathematical language by a specific sort of mathematical definition, namely by axioms. "The axioms form the fundamental vocabulary of mathematics and unambiguously determine the expressive power of mathematics." Hromkovic also makes the point that inserting new axioms into mathematics increases likewise the argumentative power of mathematics: by means of these new axioms, it becomes possible to investigate and analyze things which mathematics hitherto could not study. Thus, it is possible to reformulate Hilbert's dream in the following way: that the expressive power and the argumentative power of mathematics coincide. Gödel destroyed Hilbert's dream by proving that as a matter of fact they do not coincide, that is that the expressive power of mathematics is greater than the argumentative one, and that the gap between the two is fundamental, i.e., not eliminable. Part of Hilbert's dream was also that for each problem expressible in the language of mathematics there be a method for solving it. Turing did not share this conviction and he faced the problem of how to prove the non-existence of a mathematical method of solution for a concrete problem. To solve this problem, he had to turn the concept of a mathematical method of solution (an algorithm) into a new mathematical concept and word. Doing this was, according to Hromkovic, Alan Turing's main contribution. With the new concept of algorithm added to the language of mathematics, it was now possible to study the range and limits of automation.

Starting point of Shapiro's article is the received view, initiated by Church in 1936, that Church's Thesis (CT) is not subject to rigorous mathematical proof or refutation, since CT is the identification of an informal, intuitive notion with a formal, precisely defined one. Kleene in 1952 asserts that the intuitive notion of computability is vague and he claims that it is impossible to mathematically prove things about "vague intuitive" notions. Gödel challenged the conclusion based on the supposed vagueness of computability. Gödel's own view seems to be that there is a precise property or notion resp. somehow underlying the intuitive notion of computability, which is to be captured by an analysis and progressive sharpening of the intuitive, apparently vague one. Shapiro's main question is: "How does one *prove* that a proposed sharpening of an apparently vague notion is the uniquely correct one" capturing the precise notion which was there already? The received view referred to above was not only challenged by Gödel, but eventually by several prominent philosophers, logicians, and historians such as Gandy, Mendelson, or Sieg. Since the Gödel/Mendelson/Sieg–position generated further discussions and critical reactions, the issues relating to CT "engage some of the most fundamental questions in the philosophy of mathematics" such as: "What is it to prove something? What counts as a proof? ... What is mathematics about?" Shapiro then sets out to show that the defenders of the position that CT is

provable do not thereby mean provable by a ZFC-proof, nor could they possibly mean provable by a formal proof. Thus, he reaches the conclusion that even though CT is not entirely a formal or ZFC matter, this doesn't preclude it from being any less mathematical or from being provable. In order to defend this line of argument, he opposes the foundationalist model of mathematical knowledge to a holistic one. According to the latter model, it is possible to explain why Sieg's proof of CT might justifiably be called so, although it is neither a purely formal nor a ZFC proof. It can be called a proof because it is formally sound and because its premises are sufficiently evident. This according to Shapiro doesn't say that the proof may not beg any questions. But he concludes that at least "Sieg's discourse and, for that matter, Turing's are no more question-begging than any other deductive argumentation. The present theme is more to highlight the holistic elements that go into the choice of premises, both in deductions generally and in discourses that are rightly called "proofs," at least in certain intellectual contexts."

Soare sets off explaining what Turing's Thesis (TT) and what Gandy's Thesis M is. After characterizing the concept of a thesis as opposed to that of a statement of facts or a theorem, Soare considers the question whether Turing proved his assertion (that a function on the integers is effectively calculable iff it is computable by a so-called Turing machine) "beyond any reasonable doubt or whether it is merely a thesis, in need of continual verification." In his sketchy presentation of Turing's 1936 paper, Soare points out that in Turing's analysis of the notion of a mechanical procedure, Turing broke up the steps of such a procedure into the smallest steps, which could not be further subdivided. And when going through Turing's analysis, one is left with something very close to a Turing machine designed to carry out those elementary steps. Soare then relates on Gödel's, Church's, and Kleene's reaction to Turing's 1936 paper. In 1936, Post independently formulated an assertion analogous to Turing's, and he called it a working hypothesis. Somewhat in the same vein, Kleene in 1943 and especially in 1952 called Turing's assertion a thesis, which in the sequel led to the standard usage of "Turing's Thesis." Already in 1937, Church objected to Post's working hypothesis and in 1980 and 1988 Gandy challenged Kleene's claim that Turing's assertion is a thesis (i.e., could not be proved). Soare emphasizes that not only Gandy, but later on also Sieg, Dershowitz, and Gurevich as well as Kripke have presented proofs of Turing's assertion. Since Soare endorses those proofs, he gets to the conclusion that "Turing's Thesis" TT shouldn't be called that way any longer, it should rather be called Turing's Theorem.

Barry Cooper's aim is "to make clearer the relationship between the typing of information—a framework basic to all of Turing's work—and the computability theoretic character of emergent structures in the real universe." Cooper starts off with the question where incomputability comes from. He notes that there is no notion of incomputability without an underlying model of computation, which is here provided by the classical Turing machine model. He then observes that whereas from a logical point of view, a Turing machine is fairly simple, any embodiment of a particular universal Turing machine as an actual machine is highly non-trivial. "The physical complexities have been packaged in a logical structure, digitally coded," and "the logical view has reduced the type of the information embodied in

the computer." Cooper first considers the relationship between incomputability and randomness. The notion of randomness used to describe the lack of predictability is sometimes taken to be more intuitive than the one of incomputability. Cooper argues that randomness turns out to be a far more complicated notion, and that all that could be substantiated "building on generally accepted assumptions about physics, was incomputability." He then proceeds from the observation that "global patterns can often be clearly observed as emergent patterns in nature and in social environments, with hard to identify global connection to the underlying computational causal context." Cooper discusses this gap by reflecting on the relationship between the halting problem (one of the classical paradigms of incomputability) and the Mandelbrot set. For him, the Mandelbrot set "provides an illuminating link between the pure abstraction of the halting problem, and the strikingly embodied examples of emergence in nature." Cooper generalizes his observations about this link by introducing the mathematics of definability. The higher properties of a structure (those important to understand in the real world) are the large-scale, emergent relations of the structure, and the connection between these and their underlying local structure is mathematically formalized in terms of definability. "Such definability can be viewed as computation over higher type data." However, as Cooper explains, "computation over higher-type information cannot be expected to have the reliability or precision of the classical model." And he uses the example of the human brain and its hosting of complex mentality to illustrate this. The mathematics to be used for this new type of computation is based on the theory of degrees of incomputability (the so-called Turing degrees) and the characterization of the Turing definable relations over the structure of the Turing degrees.

We have to close this introduction with the very sad news that Barry Cooper, the author of the last article of this volume, after a brief illness passed away on October 26, 2015. Barry has been the driving force behind the Turing Centenary celebrations in 2012 and of the Computability in Europe movement since 2005. His impact and work as a supporter of Alan Turing are ubiquitous. We dedicate our volume to Barry. His gentle and enthusiastic personality will be greatly missed.

Acknowledgements

This is our opportunity to thank those people who made a special contribution to this volume or who contributed to making this a special volume. We'd like to express our great gratitude to Martin Davis who graciously took upon himself the task of writing a wonderful preface. We'd also like to express this gratitude to Jack Copeland who contributed a couple of brilliant ideas the specific nature of which remains a secret between him and us. Moreover, we are very grateful to the referees of all the articles for their refereeing job, and especially to those who did a particularly thorough job. And we'd finally like to thank Dr. Barbara Hellriegel, Clemens Heine and Katherina Steinmetz from Birkhäuser/Springer Basel, and Venkatachalam Anand from SPi

Content Solutions/SPi Global very much for not despairing about the delay with which we delivered the goods and for making up for this delay with a very efficient and very friendly high speed production process.

Zurich Giovanni Sommaruga
Bern Thomas Strahm
December 2015

Contents

Part III Philosophical Reflections

Part I
Turing and the History of Computability Theory

Conceptual Confluence in 1936: Post and Turing

Martin Davis and Wilfried Sieg

Abstract In 1936, Post and Turing independently proposed two models of computation that are virtually identical. Turing refers back to these models in his (The word problem in semi-groups with cancellation. Ann. Math. **52**, 491–505) and calls them "the logical computing machines introduced by Post and the author". The virtual identity is not to be viewed as a surprising coincidence, but rather as a natural consequence of the way in which Post and Turing conceived of the steps in mechanical procedures on finite strings. To support our view of the underlying conceptual confluence, we discuss the two 1936 papers, but explore also Post's work in the 1920s and Turing's paper (Solvable and unsolvable problems. Sci. News **31**, 7–23). In addition, we consider their overlapping mathematical work on the word-problem for semigroups (with cancellation) in Post's (Recursive unsolvability of a problem of Thue. J. Symb. Log. **12**, 1–11) and Turing's (The word problem in semi-groups with cancellation. Ann. Math. **52**, 491–505). We argue that the unity of their approach is of deep significance for the theory of computability.

Keywords Ackermann • Bernays • Canonical system • Church • Computor • Correspondence decision problem • Davis • Dedekind • Dershowitz • Gandy • Gödel • Gurevich • Herbrand • Hilbert • Kleene • Kolmogorov • Mechanical procedure • Normal system • Post • Principia mathematica • Production • Recursive functions • Representation theorem • Sieg • Skolem • Tag • Thue • Turing • Turing's central thesis • Unsolvability • Uspenski • Word problem

AMS Classifications: 01A60,03-03,03D03,03D10,03D20, 20M05

M. Davis (✉)
Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 3360 Dwight Way, Berkeley, CA 94704-2523, USA
e-mail: martin@eipye.com

W. Sieg
Department of Philosophy, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

G. Sommaruga, T. Strahm (eds.), *Turing's Revolution*,
DOI 10.1007/978-3-319-22156-4_1

1 Introduction

Princeton was an exciting place for logicians in the mid-1930s. Church had been appointed as assistant professor of mathematics in 1929 and, together with his students Stephen C. Kleene and J. Barkley Rosser, started to create a new subject soon to be called "recursive function theory".[1] Their work was propelled by von Neumann's presentation of Gödel's incompleteness theorems [27] in the fall of 1931 and Gödel's lectures "On undecidable propositions of formal mathematical systems" in the spring of 1934 [28]. Paul Bernays, the collaborator of David Hilbert on proof theory and the related foundational program, visited for the academic year 1935/1936 and gave the lectures [2]. Even before that visit, Bernays and Church had extensive correspondence concerning the ongoing work in recursion theory; see [59]. From September 1936 to July 1938, Alan Turing was studying in Princeton as Church's Ph.D. student; his dissertation "Systems of logic based on ordinals" was published in [73].

In 1936, Church founded a new quarterly devoted to contemporary research in logic: The Journal of Symbolic Logic. Its first volume contained a three-page paper by Emil L. Post, "Finite combinatory processes: Formulation I". The editors added a footnote to the paper that read:

> Received October 7, 1936. The reader should compare an article by A.M. Turing, On computable numbers, shortly forthcoming in Proceedings of the London Mathematical Society. The present article, however, although bearing a later date, was written entirely independently of Turing's.

Post was at the time teaching at City College in New York City and had also contact with Church, both personal and through correspondence.

As to the remark concerning Post's paper, it would indeed be readily apparent to any reader of Turing's and Post's articles that their basic idea of "computation" was the same. Turing presented a machine model; his machines were "supplied with a 'tape' ... divided into sections, called 'squares', each capable of bearing a 'symbol' " from a finite list, possibly just two different ones. In the latter case, we refer to the machine as a two-letter-machine.[2] Instead of a tape divided into squares, Post wrote of a "symbol space ... to consist of a two way infinite sequence of spaces or boxes"; each box could be either "marked" or "unmarked". In both conceptions, at a given instant, one particular square/box was in play, and the permitted basic operations were to change the symbol in the square/box and/or to move to an adjacent square/box. Turing imagined these steps carried out by a

[1] The broader history of computability has been described in a number of publications, partly by the participants in the early development in Princeton, for example, [43, 57]. Good discussions are found in, among others [9, 14, 26, 58, 62, 68].

There are many excellent books on recursion/computability theory, but not many that take Post's approach as fundamental. We just mention [13, 47, 56, 67].

[2] Machines that are deterministic are Turing's *a-machines*; if a-machines operate only on 0s and 1s they are called *computing machines*; see [71], p. 232.

simple mechanism with a finite number of states where each step included a possible change of state. In Post's formulation, the computation is carried out by a human "worker" who is following a fixed sequence of instructions including what we would call a conditional jump instruction. The jump instruction is also part of a program for Turing's machine: depending on the symbol in the square and its state, the machine may follow a different next instruction.

It seems astonishing that, on opposite sides of the Atlantic, two models of computation were proposed that are virtually identical. We suggest that this virtual identity is not to be viewed as a surprising coincidence, but rather as a natural consequence of the way in which Post and Turing conceived of steps in mechanical procedures on finite strings. To support our view of the underlying conceptual confluence, we discuss the two 1936 papers, but explore also Post's work in the 1920s and Turing's paper [75]. We consider in addition their overlapping mathematical work on the word problem for semigroups [54] and for semigroups with cancellation [74]. We argue that the unity of their approaches is of deep significance for the *theory* of computability.[3]

2 Substitution Puzzles: The Link

"Solvable and Unsolvable Problems", [75], is presumably Turing's last published paper. It appeared in the British journal *Science News*. The paper focuses on the methodological problems surrounding effective calculability, now usually discussed under the heading of Church's or Turing's Thesis. Turing argues in it for the adequacy of a rigorous notion of computability, but without ever mentioning his computing machines. That may be surprising: after all, for Church, Gödel, and many contemporary computer scientists, it is the analysis of effective, mechanical procedures in terms of "finite machines" that makes Turing's work so convincing. Instead, Turing uses here as the basic concept that of "unambiguous substitution puzzles"—a form of Post production systems. Post had used these special puzzles in [54] to describe Turing machines elegantly and to establish the unsolvability of the word problem for semigroups. Post's techniques were refined in [74] to extend the unsolvability result to semigroups with cancellation.

Underlying our presentation, as well as Turing's exposition in [75], is the recognition of the essential unity of Turing's *philosophical analysis of mechanical procedures* in [71] and Post's *mathematical work on normal forms* of production systems. Post's work was done during the 1920s, but published only in [51]. This unity goes far deeper than the confluence via extensional equivalences between different notions of computability for number theoretic functions as expounded so beautifully in [26]. It also goes deeper than just observing that the models of

[3]There are other commonalities in their approaches, e.g., in connection with relative computability, which first appeared in Turing's dissertation and which played such a key role in Post's later work. However, here we are focusing on their fundamental conceptual analysis.

computation Post and Turing introduced in 1936 are essentially the same. Here is Turing's retrospective assertion from the beginning of his [74], p. 491:

> The method [of proof] depends on reducing the unsolvability of the problem in question to a known unsolvable problem connected with the logical computing machines introduced by Post [49] and the author [71].

Turing points, in a dramatic way, to the structural identity of the computations of his two-letter machine and those of the worker in [49].

Post formulated no intrinsic reason for the model he presented in this paper, but he conjectured it to be equivalent to the "Gödel-Church development". If one considers also his work on canonical systems from the 1920s (discussed below), then there is an indication of a reason: the model may be viewed as a playful description of a simple production system to which canonical systems can be reduced. Turing's unambiguous substitution puzzles include these simple production systems. In order to specify such puzzles one is given an unlimited supply of "counters", possibly of only two distinct kinds. A finite sequence of counters is an initial configuration, and the puzzle task is to transform the given configuration into another one using substitutions from a fixed finite list of rules. Such a puzzle, though not by this name, is obtained in Sect. 9,I of Turing's [71] at the very end of his analysis of mechanical procedures; it can be carried out by a suitably generalized machine that operates on strings, a *string machine*, as presented in Sect. 5. A good example of a substitution puzzle, Turing asserts in [75], is "the task of proving a mathematical theorem within an axiomatic system". The abstract investigation of just this task for parts of Whitehead and Russell's *Principia Mathematica* was the starting point of Post's work in the 1920s, as we discuss in Sect. 4.

The "theorem-proving-puzzle" for first-order logic was not only Post's problem, but was also one of the central issues in mathematical logic during the 1920s: is it decidable by a mechanical procedure whether a particular given statement can be proved from some assumptions? Commenting on this problem, best known as Hilbert's *Entscheidungsproblem*, von Neumann conjectured in 1926 that it must have a negative solution and added, "we have no idea how to prove this". Ten years later Turing showed that the problem has indeed a negative solution, after having addressed first the key conceptual issue, namely, to give a precise explication of "mechanical procedure".[4] The explication by his machine model of computation is grounded in the analysis that leads, as mentioned, to a substitution puzzle. In parallel to what we said above about Post's 1936 model, Turing's two-letter machine may be viewed as a playful formulation of a machine to which string machines can provably be reduced and to which, in turn, the mechanical processes of a human computing agent can be reduced, if these processes (on strings or other concrete symbolic configurations) are constrained by finiteness and locality conditions; see Sect. 5.

[4] In the same year, Church established the unsolvability of the *Entscheidungsproblem*—having identified λ-definability and general recursiveness with effective calculability; [6, 7].

The conceptual confluence of Turing's work with Post's is rather stunning: Post's worker model and Turing's two-letter machine characterize exactly the same class of computations. The crucial link are the simple substitution puzzles (with just two counters) that are uniquely connected to each instance of both models. The wider confluence indicated above is discussed in detail in Sects. 4 and 5; its very special character is seen perhaps even more vividly when comparing it with the contemporaneous attempts to analyze directly the effective calculability of number theoretic functions.

3 Effectively Calculable Functions

The thoughts of mathematical logicians like Hilbert, Bernays, Herbrand, Gödel,... about mechanical procedures on syntactic configurations were not proceeding at all along the lines of Post or Turing, but were naturally informed by paradigmatic recursive procedures for calculating the values of number theoretic functions. Dedekind introduced primitive recursive functions in [19], Skolem used them systematically in [66] to develop an elementary part of number theory, and Hilbert and Bernays exploited during the 1920s the calculability of their values in proof theoretic investigations. In his [37], Herbrand expanded the class to "finitist" functions and included, in particular, the effectively calculable, but non-primitive recursive Ackermann function.

Gödel built on that work when introducing *general recursive functions* via his equation calculus in [28], the write-up of lectures given at the Institute for Advanced Study.[5] In the famous footnote 3 to this article, Gödel had suggested that a suitable generalization of the primitive recursive functions permitting all possible recursions would encompass the class of all effectively calculable number theoretic functions. His formulation via the equation calculus then came in the last section of the notes. However, despite appearances to the contrary, Gödel insisted in a letter to one of the authors (see [14]) that he was not prepared at that time to conclude that his formulation accomplished that goal, because until Kleene's elucidations [40], it was not clear that the notion was sufficiently robust.

Gödel's mathematically well-defined class of general recursive functions was identified in [5] with the informal class of effectively calculable number theoretic functions. Kleene labeled this identification as *Church's Thesis* in [41, 42]; the

[5] In the early evolution of recursion theory, Gödel's definition was viewed as being a modification of a proposal of Herbrand's—because Gödel presented it that way in his Princeton Lectures. In a letter to Jean van Heijenoort in 1964, Gödel reasserted that Herbrand had suggested, in a letter, a definition very close to the one actually presented in [28]. However, the connection of Gödel's definition to Herbrand's work is much less direct; that is clear from the two letters that were exchanged between Gödel and Herbrand in 1931. John Dawson found the letters in the Gödel Nachlass in 1986; see [17]. The letters are published in [36]; their intellectual context is discussed in [61].