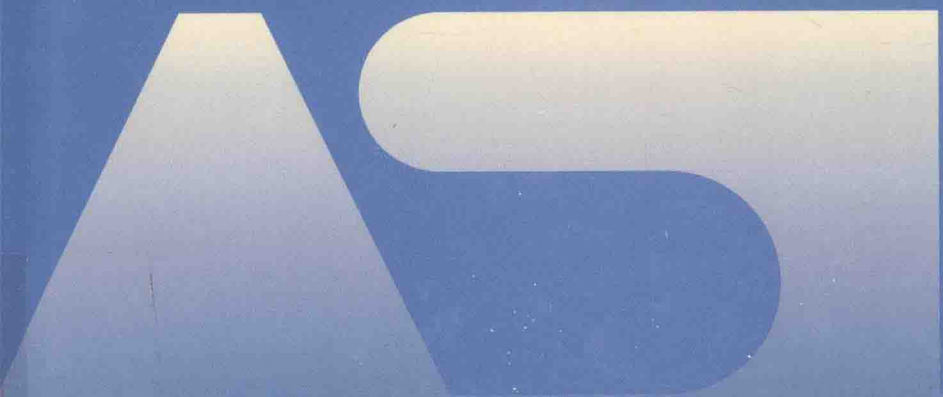
LECTURE NOTES IN LOGIC

# TURING'S LEGACY

## DEVELOPMENTS FROM TURING'S IDEAS IN LOGIC

EDITED BY
ROD DOWNEY



CAMBRIDGE

Lecture Notes in Logic 42

# Turing's Legacy: Developments from Turing's Ideas in Logic

*Edited by*
ROD DOWNEY
*Victoria University of Wellington*

ASSOCIATION FOR SYMBOLIC LOGIC

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781107043480

© Association for Symbolic Logic 2014

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2014

Printed in the United Kingdom by CPI Group Ltd, Croydon CRO 4yy

*A catalogue record for this publication is available from the British Library*

*Library of Congress Cataloging-in-Publication Data*
Turing's legacy : developments from Turing's ideas in logic / edited by Rod Downey, Victoria University of Wellington.
pages cm. – (Lecture notes in logic ; 42)
Includes bibliographical references and index.
ISBN 978-1-107-04348-0 (hardback)
1. Computational complexity. 2. Machine theory. 3. Turing, Alan Mathison, 1912-1954. I. Downey, R. G. (Rod G.), editor of compilation.
QA267.7.T87 2014
510.92–dc23 2014000240

ISBN 978-1-107-04348-0 Hardback

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this publication, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

# TURING'S LEGACY: DEVELOPMENTS FROM TURING'S IDEAS IN LOGIC

**§1. Introduction.** The year 2012 was the centenary of the birth of one of the most brilliant mathematicians of the 20th century. There were many celebrations of this fact, and many conferences based around Turing's work and life during 2012. In particular, there was a half year program (Syntax and Semantics) at the Newton Institute in Cambridge, and many "Turing 100/Centenary" conferences throughout the year. These events included truly major meetings featuring many of the world's best mathematicians and computer scientists (and even Gary Kasparov) around his actual birth day of June 23, including *The Incomputable, ACM A. M. Turing Centenary Celebration, How the World Computes* (CiE 2012), and *The Turing Centenary Conference*. There are also a number of publications devoted to Turing's life, work and legacy.

To the general public, Turing is probably best known for his part in Bletchley Park and the war-winning efforts of the code-breakers at Hut 8. To biologists, Turing is best known for his work on morphogenesis, the paper "A Chemical Basis for Morphogenesis" being his most highly cited work.

To logicians, and computer scientists, Alan Turing is best known for his work in computation, arguably leading to the development of the digital computer. This development has caused almost certainly the most profound change in human history in the last century. Turing's work in computation grew from philosophical questions in logic. Thus it seems fitting that the *Association for Symbolic Logic* sponsored this volume.

The idea for this volume is to (mainly) look at the various ways Turing's ideas in logic have developed into major programs in the landscape of mathematics and philosophy in th early 21st Century. That is, where did these ideas go? A number of leading experts were invited to participate in this enterprise. All of the papers were reviewed both for readability by non-experts and for content by other experts.

**§2. Turing's work.** There is an excellent archive of Turing's work in

`http://www.turing.org.uk/sources/biblio.html`.

Jack Copeland (sometimes with Diane Proudfoot) have historical articles and books such as [1, 2, 3]. Below we give a few brief comments and refer the

Research supported by the Marsden Fund of New Zealand.

reader to these and the books Davis [4] and to Herken [5] for more historical comments, as well as the articles in this volume by Nerode, Sieg and Soare.

Turing [7] worked famously on the *Entscheidungsproblem*, the question of the decision problem for validity of first order predicate calculus. His work and that of Church, Kleene, Post and others solved the problem. Turing's paper of 1936 laid the foundations for the advent of stored program computers. His 1936 paper had the key idea of stored program computers via universal machines. Turing knew of the possibilities of large scale electronic computers following the groundbreaking ideas of Fred Flowers with his work on *Colossus* in the second world war. Turing's analysis of computation and his introduction of universal machines are discussed in both Sieg's and Soare's articles.

In a lecture of 1947, Turing said of his design of ACE (automated computing engine)

> "The special machine may be called the universal machine; it works in the following quite simple manner. When we have decided what machine we wish to imitate we punch a description of it on the tape of the universal machine .... The universal machine has only to keep looking at this description in order to find out what it should do at each stage. Thus the complexity of the machine to be imitated is concentrated in the tape and does not appear in the universal machine proper in any way .... [D]igital computing machines such as the ACE ... are in fact practical versions of the universal machine."

In 1943, McCulloch and Pitt used Turing ideas to show the control mechanism for a TM could be simulated by a finite collection of gates with delays. These ideas were later developed by von Neumann and others which lead to ENIAC in 1943. A friend of von Newmann who worked with him on the Atomic bomb project was Stanley Frankel (see [3]) who is quoted as saying the following:

> "von Neumann was well aware of the fundamental importance of Turing's paper of 1936 'On computable numbers ... ', which describes in principle the 'Universal Computer' ... Many people have acclaimed von Neumann as the 'father of the computer' (in a modern sense of the term) but I am sure that he would never have made that mistake himself. He might well be called the midwife, perhaps, but he firmly emphasized to me, and to others I am sure, that the fundamental conception is owing to Turing."

Turing designed the ACE. Whilst never built, Turing's design was the basis of the architecture of several computers. For example, Huxley's G15 computer, the first PC (about the size of a fridge) was based on it, with about 400 sold worldwide, and remaining in use until 1970(!).

The world's first programmable computer was built in Manchester by Turing's lifelong friend Max Newman (who Turing met in 1935). Turing was

此为试读,需要完整PDF请访问: www.ertongbook.com

involved in this project and wrote the world's first programming manual. Turing also proposed methods of symbolic program verification, and logically constructing programs. His thesis, "Systems of logic based on ordinals," looked at transfinite methods of verification. In this thesis [8], Turing also introduces the notion of an oracle Turing machine which is essential for our understanding of relative computability and computational complexity.

Remarkably, Turing wrote the world's first computer chess program *before there were programmable computers*. The reader interested in this should look at Kasparov's talk video-ed at the Turing Centenary Conference at Manchester (Kasparov and Friedel [6]). Turing's ideas of using optimization of functions as a control method for artificial intelligence are at the vanguard of all such work.

Turing thought deeply about artificial intelligence with articles such as [9]. This is reflected by the well-known Turing test and the article he wrote whilst on sabbatical in Cambridge, infamously judged as a "schoolboy effort" by Charles Darwin, his government boss. For more on this see Copeland–Proudfoot [3].

The present volume has an article by Freer, Roy and Tennenbaum around AI. As well, we have a fascinating article on automated theorem proving by Hales. This article also is concerned with practical aspects of computation, something of great interest to Turing as witnessed by his famous article on ill-posedness of matrix operations. Lenore Blum has contributed a article to this volume about such questions and the "other theory of computation."

Other generalizations of the notion of algorithm are discussed in articles by Welch and by Normann. These are "higher" generalizations of the notion of computation to computations on ordinals and the transfinite. Finally, the generalization of the notion of computation to the quantum is given in Buhrman's article.

Turing's ideas of computation are critical to our understanding of randomness. The article by Downey looks at the development of pure computability theory, and its use in the theory algorithmic randomness. Downey's article also looks at Turing's anticipation of the Martin-Löf idea of using computation to bound the theory of measure as seen in his unpublished work on normality (Turing [10]).

Turing's original paper [7] was concerned with computation of the *real numbers and functions*. Thus, he wrote the first paper on computable analysis. Computable analysis and its developments are discussed in the article by Avigad and Brattka.

The article by Homer and Selman discusses how modern computational complexity theory has developed from Turing's ideas.

Turing had many technical contributions in mathematical logic. His early articles in the *Journal of Symbolic Logic* showed the equivalences of various models of computation. Turing also proved the undecidability of the word

problem for cancellation semigroups. Charles Miller III contributes a long article concerning how these ideas have panned out in the area of combinatorial group theory, and where this important subject has gone in the last 50 years. In the same spirit, the article by Fokina, Harizanov and Melnikov looks at how Turing's ideas have developed in the computability theory of structures, such as logical models and algebraic structures.

We have not concerned ourselves with Turing's work on morphogenesis, as we are concentrating on *logical* developments. For this reason we have chosen not to follow his ideas on code-breaking and cryptography. These are very well chronicled in many books on the work of Turing and others at Bletchley park. It is widely reported that the work of this group of 1,200 workers and the efforts of the powerful group of mathematicians in "Hut 8" shortened the war by at least 2 years, and saved millions of lives. It is also clear that we could have added several other articles in other areas.

Nevertheless, we believe that this volume here represents an important collection of articles giving an insight into both a great mind and into how mathematics and logic (in and about computation) have developed in the last 70 years. We hope you enjoy the result.

*The Editor*
Rod Downey

REFERENCES

[1] JACK COPELAND, ***The essential Turing***, Oxford University Press, Oxford and New York, September 2004.

[2] ———, ***Alan Turing's automatic computing engine: The master codebreaker's struggle to build the modern computer***, Oxford University Press, Oxford and New York, June 2005.

[3] JACK COPELAND and DIANE PROUDFOOT, *Alan Turing father of the modern computer*, ***The Rutherford Journal***, vol. 4 (2011–2012), `http://www.rutherfordjournal.org/article040101.html`.

[4] MARTIN DAVIS, ***The undecidable: Basic papers on undecidable propositions, unsolvable problems and computable functions***, Dover, 1965.

[5] ROLF HERKEN, ***The universal Turing machine: A half-century survey***, Springer-Verlag, 1995.

[6] GARY KASPAROV and FREDERIC FRIEDEL, *The reconstruction of Turing's "Paper Machine"*.

[7] ALAN TURING, *On computable numbers with an application to the Entscheidungsproblem*, ***Proceedings of the London Mathematical Society***, vol. 42 (1936), pp. 230–265, correction in ***Proceedings of the London Mathematical Society*** vol. 43 (1937), pp. 544–546.

[8] ———, *Systems of logic based on ordinals*, ***Proceedings of the London Mathematical Society***, vol. 45 (1939), no. 2, pp. 161–228.

[9] ———, *Computing machinery and intelligence*, ***Mind***, vol. 59 (1950), pp. 433–460.

[10] ———, *A note on normal numbers*. ***Collected works of A. M. Turing: Pure mathematics*** (J. L. Britton, editor), North Holland, Amsterdam, 1992, pp. 117–119, with notes of the editor in 263–265.

# CONTENTS

# COMPUTABILITY AND ANALYSIS: THE LEGACY OF ALAN TURING

JEREMY AVIGAD AND VASCO BRATTKA

§1. **Introduction.** For most of its history, mathematics was algorithmic in nature. The geometric claims in Euclid's *Elements* fall into two distinct categories: "problems," which assert that a construction can be carried out to meet a given specification, and "theorems," which assert that some property holds of a particular geometric configuration. For example, Proposition 10 of Book I reads "To bisect a given straight line." Euclid's "proof" gives the construction, and ends with the (Greek equivalent of) Q.E.F., for *quod erat faciendum*, or "that which was to be done." Proofs of theorems, in contrast, end with Q.E.D., for *quod erat demonstrandum*, or "that which was to be shown"; but even these typically involve the construction of auxiliary geometric objects in order to verify the claim.

Similarly, algebra was devoted to developing algorithms for solving equations. This outlook characterized the subject from its origins in ancient Egypt and Babylon, through the ninth century work of al-Khwarizmi, to the solutions to the quadratic and cubic equations in Cardano's *Ars Magna* of 1545, and to Lagrange's study of the quintic in his *Réflexions sur la résolution algébrique des équations* of 1770.

The theory of probability, which was born in an exchange of letters between Blaise Pascal and Pierre de Fermat in 1654 and developed further by Christian Huygens and Jakob Bernoulli, provided methods for calculating odds related to games of chance. Abraham de Moivre's 1718 monograph on the subject was entitled *The Doctrine of Chances: or, a Method for Calculating the Probabilities of Events in Play*. Pierre de Laplace's monumental *Théorie analytique des probabilités* expanded the scope of the subject dramatically, addressing statistical problems related to everything from astronomical measurement to the measurement in the social sciences and the reliability of testimony. Even so, the emphasis remained fixed on explicit calculation.

Analysis had an algorithmic flavor as well. In the early seventeenth century, Cavalieri, Fermat, Pascal, and Wallis developed methods of computing "quadratures," or areas of regions bounded by curves, as well as volumes. In the hands of Newton, the calculus became a method of explaining and

**Turing's Legacy: Developments from Turing's Ideas in Logic**
Edited by Rod Downey
Lecture Notes in Logic, 42
© 2014, Association for Symbolic Logic

predicting the motion of heavenly and sublunary objects. Euler's *Introductio in Analysis Infinitorum* of 1748 was the first work to base the calculus explicitly on the notion of a *function*; but, for Euler, functions were given by piecewise analytic expressions, and once again, his focus was on methods of calculation.

All this is not to say that all the functions and operations considered by mathematicians were computable in the modern sense. Some of Euclid's constructions involve a case split on whether two points are equal or not, and, similarly, Euler's piecewise analytic functions were not always continuous. In contrast, we will see below that functions on the reals that are computable in the modern sense are necessarily continuous. And even though Euler's work is sensitive to the rates of convergence of analytic expressions, these rates were not made explicit. But these are quibbles, and, generally speaking, mathematical arguments through the eighteenth century provided informal algorithms for finding objects asserted to exist.

The situation changed dramatically in the nineteenth century. Galois' theory of equations implicitly assumed that all the roots of a polynomial exist *somewhere*, but Gauss' 1799 proof of the fundamental theorem of algebra, for example, did not show how to compute them. In 1837, Dirichlet considered the example of a "function" from the real numbers to the real numbers which is equal to 1 on the rationals and 0 on the irrationals, without pausing to consider whether such a function is calculable in any sense. The Bolzano–Weierstraß Theorem, first proved by Bolzano in 1817, asserts that any bounded sequence of real numbers has a convergent subsequence; in general, there will be no way of computing such a subsequence. Riemann's proof of the open mapping theorem was based on the Dirichlet principle, an existence principle that is not computationally valid. Cantor's work on the convergence of Fourier series led him to consider transfinite iterations of point-set operations, and, ultimately, to develop the abstract notion of set.

Although the tensions between conceptual and computational points of view were most salient in analysis, other branches of mathematics were not immune. For example, in 1871, Richard Dedekind defined the modern notion of an *ideal* in a ring of algebraic integers, and defined operations on ideals in a purely extensional way. In other words, the operations were defined in such a way that they do not presuppose any particular representation of the ideals, and definitions do not indicate how to compute the operations in terms of such representations. More dramatically, in 1890, Hilbert proved what is now known as the Hilbert Basis Theorem. This asserts that, given any sequence $f_1, f_2, f_3, \ldots$ of multivariate polynomials over a Noetherian ring, there is some $n$ such that for every $m \geq n$, $f_m$ is in the ideal generated by $f_1, \ldots, f_n$. Such an $m$ cannot be computed by surveying elements of the sequence, since it is not even a continuous function on the space of sequences; even if a sequence $x, x, x, \ldots$ starts out looking like a constant sequence, one cannot rule out the possibility that the element 1 will eventually appear.

此为试读,需要完整PDF请访问: www.ertongbook.com

Such shifts were controversial, and raised questions as to whether the new, abstract, set-theoretic methods were appropriate to mathematics. Set-theoretic paradoxes in the early twentieth century raised the additional question as to whether they are even consistent. Brouwer's attempt, in the 1910s, to found mathematics on an "intuitionistic" conception raised a further challenge to modern methods, and in 1921, Hermann Weyl, Hilbert's best student, announced that he was joining the Brouwerian revolution. The twentieth century *Grundlagenstreit*, or "crisis of foundations," was born.

At that point, two radically different paths were open to the mathematical community:

- Restrict the methods of mathematics so that mathematical theorems have direct computational validity. In particular, restrict methods so that sets and functions asserted to exist are computable, as well as infinitary mathematical objects and structures more generally; and also ensure that quantifier dependences are also constructive, so that a "forall-exists" statement asserts the existence of a computable transformation.
- Expand the methods of mathematics to allow idealized and abstract operations on infinite objects and structures, without concern as to how these objects are represented, and without concern as to whether the operations have a direct computational interpretation.

Mainstream contemporary mathematics has chosen decisively in favor of the latter. But computation is important to mathematics, and faced with a nonconstructive development, there are options available to those specifically interested in computation. For example, one can look for computationally valid versions of nonconstructive mathematical theorems, as one does in computable and computational mathematics, constructive mathematics, and numerical analysis. There are, in addition, various ways of measuring the extent to which ordinary theorems fail to be computable, and characterizing the data needed to make them so.

With Turing's analysis of computability, we now have precise ways of saying what it means for various types of mathematical objects to be computable, stating mathematical theorems in computational terms, and specifying the data relative to which operations of interest are computable. Section 2 thus discusses *computable analysis*, whereby mathematical theorems are made computationally significant by stating the computational content explicitly.

There are still communities of mathematicians, however, who are committed to developing mathematics in such a way that every concept and assertion has an *implicit* computational meaning. Turing's analysis of computability is useful here, too, in a different way: by representing such styles of mathematics in formal axiomatic terms, we can make this implicit computational interpretation mathematically explicit. Section 3 thus discusses different styles of constructive mathematics, and the computational semantics thereof.

## §2. Computable analysis.

**2.1. From Leibniz to Turing.** An interest in the nature of computation can be found in the seventeenth century work of Leibniz (see [47]). For example, his *stepped reckoner* improved on earlier mechanical calculating devices like the one of Pascal. It was the first calculating machine that was able to perform all four basic arithmetical operations, and it earned Leibniz an external membership of the British Royal Society at the age of 24. Leibniz's development of calculus is better known, as is the corresponding priority dispute with Newton. Leibniz paid considerable attention to choosing notations and symbols carefully in order to facilitate calculation, and his use of the integral symbol $\int$ and the d symbol for derivatives have survived to the present day. Leibniz's work on the binary number system, long before the advent of digital computers, is also worth mentioning. A more important contribution to the study of computation was his notion of a *calculus ratiocinator*, that is, a calculus of reasoning. Such a calculus, Leibniz held, would allow one to resolve disputes in a purely mathematical fashion:[1]

> The only way to rectify our reasonings is to make them as tangible as those of the Mathematicians, so that we can find our error at a glance, and when there are disputes among persons, we can simply say: Let us calculate, without further ado, to see who is right.

His attempts to develop such a calculus amount to an early form of symbolic logic.

With this perspective, it is not farfetched to see Leibniz as initiating a series of developments that culminate in Turing's work. Norbert Wiener has described the relationship in the following way [231]:

> The history of the modern computing machine goes back to Leibniz and Pascal. Indeed, the general idea of a computing machine is nothing but a mechanization of Leibniz's *calculus ratiocinator*. It is, therefore, not at all remarkable that the theory of the present computing machine has come to meet the later developments of the algebra of logic anticipated by Leibniz. Turing has even suggested that the problem of decision, for any mathematical situation, can always be reduced to the construction of an appropriate computing machine.

**2.2. From Borel to Turing.** Perhaps the first serious attempt to express the mathematical concepts of a computable real number and a computable function on the real numbers were made by Émil Borel around 1912, the year that Alan Turing was born. Borel defined computable real numbers as follows:[2]

---

[1] Leibniz, *The Art of Discovery*, 1685 [232].

[2] All citations of Borel are from [19], which is a reprint of [18]. The translations here are by the authors of this article; obvious mistakes in the original have been corrected.

> We say that a number $\alpha$ is computable if, given a natural number $n$, we can obtain a rational number that differs from $\alpha$ by at most $\frac{1}{n}$.

Of course, before the advent of Turing machines or any other formal notion of computability, the meaning of the phrase "we can obtain" remained vague. But Borel provided the following additional information in a footnote to that phrase:

> I intentionally leave aside the practical length of operations, which can be shorter or longer; the essential point is that each operation can be executed in finite time with a safe method that is unambiguous.

This makes it clear that Borel had an intuitive notion of an algorithm in mind. Borel then indicated the importance of number representations, and argued that decimal expansions have no special theoretical value, whereas continued fraction expansions are not invariant under arithmetic operations and hence of no practical value. He went on to discuss the problem of determining whether two real numbers are equal:

> The first problem in the theory of computable numbers is the problem of equality of two such numbers. If two computable numbers are unequal, this can obviously be noticed by computing both with sufficient precision, but in general it will not be known *a priori*. One can make clear progress in determining a lower bound on the difference of two computable numbers, whose definitions satisfy known conditions.

In modern terms, what Borel seems to recognize here is that although there is no algorithm that decides whether two computable real numbers are equal, the inequality relation between computable reals is, at least, computably enumerable. He then discussed a notion of the *height* of a number, which is based on counting the number of steps needed to construct that number, in a certain way. This concept can be seen as an early forerunner of the concept of Kolmogorov complexity. Borel considered ways that this concept might be utilized in addressing the equality problem.

In another section of his paper, Borel discussed the concept of a computable real number function, which he defined as follows:

> We say that a function is computable if its value is computable for any computable value of the variable. In other words, if $\alpha$ is a computable number, one has to know how to compute the value of $f(\alpha)$ with precision $\frac{1}{n}$ for any $n$. One should not forget that, by definition, to be given a computable number $\alpha$ just means to be given a method to obtain an arbitrary approximation to $\alpha$.

It is worth noting that Borel only demanded computability at computable inputs in his definition. His definition is vague in the sense that he did not

indicate whether he had in mind an algorithm that transfers a *method* to compute $\alpha$ into a *method* to compute $f(\alpha)$ (which would later become known as *Markov computability* or the *Russian approach*) or whether he had in mind an algorithm that transfers an *approximation* of $\alpha$ into an *approximation* of $f(\alpha)$ (which is closer to what we now call a computable function on the real numbers, under the *Polish approach*). He also did not say explicitly that his algorithm to compute $f$ is meant to be uniform, but this seems to be implied by his subsequent observation:

> A function cannot be computable, if it is not continuous at all computable values of the variable.

A footnote to this observation then indicates that he had the Polish approach in mind:[3]

> In order to make the computation of a function effectively possible with a given precision, one additionally needs to know the modulus of continuity of the function, which is the [ ... ] relation [ ... ] between the variation of the function values with respect to the variation of the variable.

Borel went on to discuss different types of discontinuous functions, including those that we now call Borel measurable. In fact, the entire discussion of computable real numbers and computable real functions is preliminary to Borel's development of measure theory, and the discussion was meant to motivate aspects of that development.

**2.3. Turing on computable analysis.** Turing's landmark 1936 paper [207] is titled "On computable numbers, with an application to the Entscheidungsproblem." It begins as follows:

> The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integrable variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and

[3]Hence Borel's result that computability implies continuity *cannot* be seen as an early version of the famous theorem of Ceĭtin, in contrast to what is suggested in [214]. The aforementioned theorem states that any Markov computable function is already effectively continuous on computable inputs and hence computable in Borel's sense, see sections 2.5 and 3, Figure 5. While this is a deep result, the observation that computable functions in Borel's sense are continuous is obvious.

> so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

At least two things are striking about these opening words. The first is that Turing chose not to motivate his notion of computation in terms of the ability to characterize the notion of a computable function from $\mathbb{N}$ to $\mathbb{N}$, or the notion of a computable *set* of natural numbers, as in most contemporary presentations: but, rather, in terms of the ability to characterize the notion of a computable real number. In fact, Section 10 is titled "Examples of large classes of numbers which are computable." There, he introduced the notion of a computable function on computable real numbers, and the notion of computable convergence for a sequence of computable numbers, and so on; and argued that, for example, $e$ and $\pi$ and the real zeros of the Bessel functions are computable. The second striking fact is that he also flagged his intention of developing a full-blown theory of computable real analysis. As it turns out, this was a task that ultimately fell to his successors, as we explain below.

The precise definition of a computable real number given by Turing in the original paper can be expressed as follows: a real number $r$ is *computable* if there is a computable sequence of 0s and 1s with the property that the fractional part of $r$ is equal to the real number obtained by prefixing that sequence with a binary point. There is a slight problem with this definition, however, which Turing discussed in a later correction [208], published in 1937. Suppose we have a procedure that, for every $i$, outputs a rational number $q_i$ with the property that

$$|r - q_i| < 2^{-i}.$$

Intuitively, in that case, we would also want to consider $r$ to be a computable real number, because we can compute it to any desired accuracy. In fact, it is not hard to show that this second definition coincides with the first: a real number has a computable binary expansion if and only if it is possible to compute it in the second sense. In other words, the two definitions are extensionally equivalent.

The problem, however, is that it is not possible to pass *uniformly* between these two representations, in a computable way. For example, suppose a procedure of the second type begins to output the sequence of approximations $\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \ldots$. Then it is difficult to determine what the first binary digit is, because at some point the output could jump just above or just below $\frac{1}{2}$. This intuition can be made precise: allowing the sequence in general to depend on the halting behavior of a Turing machine, one can show that there is no algorithmic procedure which, given a description of a Turing machine describing a real number by a sequence of rational approximations, computes

the digits of $r$. On the other hand, for any fixed description of the first sort, there is a computable description of the second sort: either $r$ is a dyadic rational, which is to say, it has a finite binary expansion; or waiting long enough will always provide enough information to determine the digits. So the difference only shows up when one wishes to talk about computations which take descriptions of real numbers as input. In that case, as Turing noted, the second type of definition is more natural; and, as we will see below, these are the *descriptions* of the computable reals that form the basis for computable analysis.

Turing's second representation of computable reals, presented in his correction [208], is given by the formula

$$(2i-1)n + \sum_{r=1}^{\infty} (2c_r - 1)\left(\frac{2}{3}\right)^r,$$

where $i$ and $n$ provide the integer part of the represented number and the binary sequence $c_r$ the fractional part. This representation is essentially what has later been called a *signed-digit representation* with base $\frac{2}{3}$. It is interesting that Turing acknowledged Brouwer's influence (see also [72]):

> This use of overlapping intervals for the definition of real numbers is due originally to Brouwer.

In the 1936 paper, in addition to discussing individual computable real numbers, Turing also defined the notion of a computable function on real numbers. Like Borel, he adopted the standpoint that the input to such a function needs to be computable itself:

> We cannot define general computable functions of a real variable, since there is no general method of describing a real number, but we can define a computable function of a computable variable.

A few years later Turing introduced oracle machines [210], which would have allowed him to handle computable functions on *arbitrary* real inputs, simply by considering the input as an oracle given from outside and not as being itself computed in some specific way. But the 1936 definition ran as follows. First, Turing extended his definition of computable real numbers $\gamma_n$ from the unit interval to all real numbers using the formula $\alpha_n = \tan(\pi(\gamma_n - \frac{1}{2}))$. He went on:

> Now let $\varphi(n)$ be a computable function which can be shown to be such that for any satisfactory[4] argument its value is satisfactory. Then the function $f$, defined by $f(\alpha_n) = \alpha_{\varphi(n)}$, is a computable function and all computable functions of a computable variable are expressible in this form.

[4]Turing calls a natural number $n$ *satisfactory* if, in modern terms, $n$ is a Gödel index of a total computable function.

此为试读,需要完整PDF请访问: www.ertongbook.com