Boston Studies in the Philosophy and History of Science 324

Juliet Floyd
Alisa Bokulich *Editors*

Philosophical Explorations of the Legacy of Alan Turing

Turing 100

Springer

Boston Studies in the Philosophy and History of Science

Volume 324

Editors
Alisa Bokulich, Boston University
Robert S. Cohen, Boston University
Jürgen Renn, Max Planck Institute for the History of Science
Kostas Gavroglu, University of Athens

Managing Editor
Lindy Divarci, Max Planck Institute for the History of Science

Editorial Board
Theodore Arabatzis, University of Athens
Heather E. Douglas, University of Waterloo
Jean Gayon, Université Paris 1
Thomas F. Glick, Boston University
Hubert Goenner, University of Goettingen
John Heilbron, University of California, Berkeley
Diana Kormos-Buchwald, California Institute of Technology
Christoph Lehner, Max Planck Institute for the History of Science
Peter McLaughlin, Universität Heidelberg
Agustí Nieto-Galan, Universitat Autònoma de Barcelona
Nuccio Ordine, Universitá della Calabria
Sylvan S. Schweber, Harvard University
Ana Simões, Universidade de Lisboa
John J. Stachel, Boston University
Baichun Zhang, Chinese Academy of Science

More information about this series at http://www.springer.com/series/5710

Juliet Floyd • Alisa Bokulich
Editors

Philosophical Explorations of the Legacy of Alan Turing

Turing 100

Springer

Editors
Juliet Floyd
Department of Philosophy
Boston University
Boston, MA, USA

Alisa Bokulich
Department of Philosophy
Boston University
Boston, MA, USA

ISSN 0068-0346 ISSN 2214-7942 (electronic)
Boston Studies in the Philosophy and History of Science
ISBN 978-3-319-53278-3 ISBN 978-3-319-53280-6 (eBook)
DOI 10.1007/978-3-319-53280-6

Library of Congress Control Number: 2017936968

© Springer International Publishing AG 2017
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

This volume is dedicated to the memory of two of our participants in the "Turing 100" session of the Boston Colloquium for Philosophy of Science,

Marvin Minsky (1927–2016)
and
S. Barry Cooper (1943–2015)

who, like Turing, have contributed so much to these topics and whose insights and presence will be greatly missed.

Preface

On November 11 and 12 of 2012, we celebrated the centenary of Alan Turing's (1912–1954) birth with a session of the Boston Colloquium for Philosophy of Science titled "Turing 100," which was co-sponsored by the Center for Philosophy and History of Science and the Rafik B. Hariri Institute for Computing and Computational Science and Engineering, both centers of excellence here at Boston University. We invited a distinguished team of philosophers, mathematicians, logicians, historians, computer scientists, and cryptographers to our event—among them four Turing Award winners, the highest honor now given in computer science: Marvin Minsky (1969), Michael O. Rabin (1976), Ron Rivest (2002), and Silvio Micali (2012). Other speakers included Juliette Kennedy, Martin Davis, Gerald Sacks, Theodore Slaman, Craig Bauer, Patrick Henry Winston, S. Barry Cooper, Wilfrid Sieg, Stephen Wolfram, Rohit Parikh, Mark Hogarth, Rod Downey, and Leonid Levin. Appropriately, each speaker reflected on foundational issues with power and imagination in Turing's spirit, representing a number of different fields in which speculation, rigorous mathematics, good sense, and simplicity are valued. A record of their talks and slides may be found at http://www.bu.edu/cphs/colloquium/2012–2013/#turing. "Turing 100" attracted a large audience and generated a lively discussion, closing out the Alan Turing Year in style.

A host of meetings took place around the world in 2012, which was known to users of the Internet as "The Alan Turing Year." Several recent volumes have appeared, or are slated to appear, aiming to bring the importance of Turing's ideas and life to a wider, more variegated public. Our Boston University "Turing 100" meeting was distinctive, however, in addressing itself specifically to the *philosophical* legacy of Turing's work. We asked our participants to consider foundational motives and methods, as well as look forward to the frontiers of the more applied philosophical regions of Turing's ideas. Turing counts as one of the twentieth century's most important scientists—if not one of the most influential scientists of all time—and we wanted to explore his philosophical sensibility and contributions to general issues of philosophical importance. Philosophy, and the philosophy of science more specifically, often plays a central role when it comes to revolutionary periods in science, and Turing is like many of the other great scientists in this regard.

The purpose of our volume is to show how Turing's distinctive way of thinking promoted a panoply of ideas and outstanding questions crucial to the history and philosophy of science.

In discussing Turing's philosophical legacy, we must also note the important moral and political dimensions of his life and death. Turing died in 1954, age 41, apparently by suicide, having chosen to undergo chemical castration to correct his homosexuality rather than face a year in prison for "gross indecency." Though it may not, in the end, surprise us, it must surely concern us that an ostensibly democratically liberal state claiming to have been fighting the tyranny of dictatorship would willingly—and legally—so persecute one of its most able scientists and protectors. In the fall of 2008, an Internet campaign generated the then-largest petition for a pardon in the history of Britain; a royal pardon was issued on Christmas Eve, 2013. But this was only the beginning. On February 25, 2015, the day after the Oscar Award Ceremony for the film "The Imitation Game" (2014), Turing's great-nephew and great-niece presented a petition signed by even more people—half a million on change.org—demanding the pardon of 49,000 men unjustly convicted of gross indecency before 2003 (when homosexuality was fully legalized in Britain). Turing's grand-niece put it perfectly: "It is illogical that my great uncle has been the only one to be pardoned when so many were convicted of the same crime." In February 2017 "Turing's Law" passed in the United Kingdom, automatically pardoning deceased men—including Turing himself—convicted for consensual same-sex relations before homosexuality was decriminalised, and opening up procedures for men living with convictions to apply to have their names cleared. It is a true irony of history that through the Internet itself—using devices instantiating Turing's very idea of a "Turing machine"—the sense of justice of hundreds of thousands of offended individuals could be pooled to such positive effect in Turing's name.

We think this volume is a fitting and unique tribute to the breadth and depth of Turing's philosophical side and a strong and stimulating contribution to our long-standing Springer series, *Boston Studies in the Philosophy and History of Science*.

Boston, MA, USA
March 2015

Juliet Floyd
Alisa Bokulich

Acknowledgments

We received generous funding for “Turing 100” from our two Boston University co-sponsors: the Center for Philosophy and History of Science and the Rafik B. Hariri Institute for Computing and Computational Science and Engineering. They are two truly interdisciplinary centers of intellectual excellence, one with a long tradition and the other a promising new one: we are grateful for their support.

In addition to offering us financial support, Program Committee members Azer Bestavros and Steven Homer offered especially sage advice and helped us to attract a truly distinguished speaker list. We are grateful to the Department of Computer Science at Boston University for lending us their hands.

Finally we are grateful to several talented graduate students who helped us with the manuscript at various stages along the way. In particular, we are grateful to Katherine Valde for proofreading the entire manuscript before submission, Jared Henderson for managing timely conversion of documents from LaTex to Word, and Kurt Blankschaen for helping us with the index. We express our final thanks to the Department of Philosophy at Boston University for supporting this good work of our graduate students.

Contents

1 Introduction 1
Juliet Floyd

Part I Logic and Mathematics to Philosophy

2 Turing, the Mathematician 39
Daniele Mundici and Wilfried Sieg

3 Turing, Gödel and the "Bright Abyss" 63
Juliette Kennedy

4 Justified True Belief: Plato, Gettier, and Turing 93
Rohit Parikh and Adriana Renero

5 Turing on "Common Sense": Cambridge Resonances 103
Juliet Floyd

Part II The Universal Machine: From Music to Morphogenesis

6 Universality Is Ubiquitous 153
Martin Davis

7 The Early History of Voice Encryption 159
Craig Bauer

8 Turing and the History of Computer Music 189
B. Jack Copeland and Jason Long

9 Exploring the Frontiers of Computation: Measurement Based Quantum Computers and the Mechanistic View of Computation 219
Armond Duwell

10 Embodying Computation at Higher Types 233
S. Barry Cooper

Part III Human, Machine, and Mind

11 On Computing Machinery and Intelligence 265
Patrick Henry Winston

12 From Ockham to Turing – and Back Again 279
Michael Rescorla

13 Turing and Free Will: A New Take on an Old Debate 305
Diane Proudfoot

14 Turing on the Integration of Human and Machine Intelligence 323
Susan G. Sterrett

15 Is There a Church-Turing Thesis for Social Algorithms? 339
Rohit Parikh

Index 359

About the Editors and Authors

About the Editors

Juliet Floyd is professor of philosophy at Boston University and researches the interplay between logic, mathematics, and philosophy in the late nineteenth and early twentieth centuries. She has written extensively on Wittgenstein, Gödel, and Turing and also published articles on Kant, aesthetics, early analytic philosophy and American pragmatism. She is currently associate senior editor in twentieth-century philosophy at the *Stanford Encyclopedia of Philosophy* and has co-edited (with S. Shieh) *Future Pasts: The Analytic Tradition in Twentieth-Century Philosophy* (Oxford, 2001) and (with J.E. Katz) *Philosophy of Emerging Media: Understanding, Appreciation, Application* (Oxford, 2016).

Alisa Bokulich is professor of philosophy at Boston University and director of the Center for Philosophy & History of Science, where she organizes the Boston Colloquium for Philosophy of Science. She is associate member of Harvard University's Department of History of Science and a series editor for *Boston Studies in the Philosophy and History of Science*. Her research focuses on issues in philosophy of science, including history and philosophy of quantum mechanics, and philosophy of the geosciences. She is author of the book *Reexamining the Quantum-Classical Relation: Beyond Reductionism and Pluralism* (Cambridge UP, 2008) and co-editor of three additional books.

About the Authors

Craig Bauer Craig Bauer was the 2011–2012 scholar-in-residence with the National Security Agency's Center for Cryptologic History. He is the editor-in-chief of *Cryptologia*, a quarterly journal devoted to all aspects of cryptology (history, mathematics, pedagogy, etc.) and the author of the book *Secret History: The Story of Cryptology*. His next book *Unsolved! History's Greatest Ciphers* is under contract with Princeton University Press. Craig enjoys lecturing to all sorts of groups, including professional mathematicians, undergraduates, high school students, and general audiences with (initially) no interest in mathematics or its history.

S. Barry Cooper was professor of mathematical logic at the University of Leeds. He graduated from the University of Oxford, and his latest research followed that of Alan Turing in its focus on the nature of mental and physical computation. He authored and edited numerous books, including *Computability Theory*, *New Computational Paradigms*, and *Computability in Context*, and he is a leading advocate of multidisciplinary research at the interface between what is known to be computable and theoretical and practical incomputability. He was chair of the Turing Centenary Committee, which coordinated the international Turing Centenary celebrations. He was president of the Association of Computability in Europe, which is responsible for the largest computability-themed international conference series, and chaired the editorial board of its Springer book series *Theory and Applications of Computability*.

B. Jack Copeland FRS NZ is distinguished professor in humanities at the University of Canterbury, New Zealand, where he is director of the Turing Archive for the History of Computing. He is also honorary research professor of philosophy at the University of Queensland, Australia, and in 2012 was Royden B. Davis visiting chair of interdisciplinary studies in the Department of Psychology at Georgetown University, Washington DC. His books include a highly accessible biography *Turing: Pioneer of the Information Age*, also *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*, *The Essential Turing*, and *Alan Turing's Electronic Brain* (all with Oxford University Press). He has published more than 100 articles on the philosophy, history, and foundations of computing and on mathematical and philosophical logic. Copeland is the recipient of the 2016 Covey Award, recognizing a substantial record of innovative research in the field of computing and philosophy.

Martin Davis Born in 1928, Davis studied with Emil Post at City College in New York and with Alonzo Church in Princeton. He is known for his work on automated deduction and on Hilbert's tenth problem and is a professor emeritus at New York University. He and his wife of 63 years now live near the campus of the University of California at Berkeley where he is a visiting scholar. Davis's book *The Universal Computer: The Road from Leibniz to Turing* was updated and reissued in 2012 for Turing's centenary.

Armond Duwell is an associate professor of philosophy at the University of Montana. He received his Ph.D. from the Department of History and Philosophy of Science at the University of Pittsburgh in 2004. He was a member of the Probability, Philosophy, and Modeling Research Group at the University of Konstanz and a visiting fellow at the Center for Philosophy of Science in Pittsburgh. His work focuses primarily on foundational issues in quantum information theory and quantum computing, but he also has interests in realism, representation, explanation, and understanding.

Juliette Kennedy is associate professor in the Department of Mathematics and Statistics at the University of Helsinki. Her work in mathematical logic is focused mainly in the areas of set-theoretic model theory and set theory. In the area of history and foundations of mathematics, she has worked extensively on a project that attempts to put Kurt Gödel in the right perspective, historically and foundationally. Her edited collection *Interpreting Gödel* (Cambridge University Press) appeared in 2014.

Jason Long is a New Zealand composer and performer, focusing on musical robotics and electro-acoustic music. He has carried out research at the Utrecht Higher School of the Arts in the Netherlands and at Tokyo University of the Arts in Japan, as well as at several universities in New Zealand. His work First Contact has been featured at festivals such as the International Society for Contemporary Music in Brussels and the International Computer Music Conference in Perth, Australia. Other works, including his Glassback and The Subaquatic Voltaic, have been performed at the Manila Composers' Lab in the Philippines and at Asian Composers League Festivals in Taipei and Tokyo, as well as in New Zealand and elsewhere.

Daniele Mundici is professor of mathematical logic in the Department of Mathematics and Computer Science, University of Florence. He is a member of the International Academy of Philosophy of Science (AIPS), Brussels, and a corresponding member of the National Academy of Exact, Physical and Natural Sciences (ANCEFN), Buenos Aires. He has been president of the Kurt Gödel Society (KGS), Vienna, from 1993 to 1996 and of the Italian Association of Logic and Applications (AILA) from 1994 to 1998. He is managing editor of several international journals in mathematical logic and algebra and their applications and the author of 5 books and over 170 papers on mathematical logic, theoretical computer science, algebra, and functional analysis.

Rohit Parikh All of Rohit Parikh's degrees are from Harvard, and his logic education came from Burton Dreben and W.V. Quine, both in the philosophy department at Harvard, and from Hartley Rogers (MIT). He has taught at Stanford University, Panjab University, Bristol University, and Boston University, before moving to the City University of New York where he has been distinguished professor (computer science, mathematics, philosophy) since 1982. His principal interest these days is in applying epistemic logic and game theory to understand social algorithms (aka social software).

Diane Proudfoot is professor of philosophy at the University of Canterbury, New Zealand. She was educated at the University of Edinburgh, Dartmouth College, the University of California at Berkeley, and the University of Cambridge. Diane has held various scholarships and visiting fellowships, including at MIT, New York University, Georgetown University, the Israel Institute for Advanced Studies, and the Swiss Federal Institute of Technology. She and Jack Copeland founded the online Turing Archive for the History of Computing and received a grant from the Royal Society of New Zealand for research into the philosophical foundations of cognitive and computer science. Diane has published in the *Journal of Philosophy*, *Artificial Intelligence*, *Scientific American*, and numerous other philosophy and science journals.

Adriana Renero is a Ph.D. candidate in philosophy at The Graduate Center, City University of New York. She received her M.A. in philosophy from the National Autonomous University of Mexico, after earning the National Award for the Best Bachelor's Thesis in Philosophy. Renero works in philosophy of mind, philosophy of cognitive science, epistemology, and philosophy of music. Her publications include "Consciousness and Mental Qualities for Auditory Sensations" and "Noûs and Aisthēsis: Two Cognitive Faculties in Aristotle." Renero's dissertation develops a novel theory of introspection. Other current projects include a paper on similarities and differences of visual and auditory perception.

Michael Rescorla is a professor in the philosophy department at the University of California, Los Angeles. His research concerns the philosophies of mind, language, and logic.

Wilfried Sieg is Patrick Suppes professor of philosophy at Carnegie Mellon and fellow of the American Academy of Arts and Sciences. He is internationally known for work in proof and computation theory, history of modern logic and mathematics, and philosophy of mathematics. A collection of essays joining these aspects of his research was published under the title *Hilbert's Programs and Beyond*. As co-director of the *Laboratory of Symbolic and Educational Computing*, he has pursued his AProS Project, pioneering strategic automated search for natural deduction proofs in logic and elementary set theory; this work is used in web-based logic courses.

Susan G. Sterrett is Curtis D. Gridley distinguished professor of history and philosophy of science at Wichita State University in Kansas, USA. She earned a B.S. from the College of Engineering at Cornell University, an M.A. in mathematics, and an M.A. and Ph.D. in philosophy from the University of Pittsburgh, after which she taught at Duke University and Carnegie Mellon University. She is the author of numerous works in logic and philosophy of science; she has published several papers on Turing's work, including "Turing's Two Tests for Intelligence," "Too Many Instincts: Contrasting Philosophical Views on Intelligence in Humans and Non-humans," and "Bringing Up Turing's 'Child-Machine'."

Patrick Winston is Ford professor of artificial intelligence and MacVicar faculty fellow at the Massachusetts Institute of Technology. A graduate of MIT, his Genesis Research Group focuses on developing a computational account of human intelligence and how it differs from that of other species, with special attention to modeling human storytelling and comprehension. He is the author and editor of numerous books, including *Artificial Intelligence*; he served as director of the MIT Artificial Intelligence Laboratory (now part of MIT's CSAIL Laboratory) for 25 years. He is now research coordinator for the multiuniversity, multidisciplinary Center for Brains, Minds and Machines centered at MIT.

Chapter 1
Introduction

Juliet Floyd

Abstract An overview of essays in this volume, with an emphasis on the philosophical legacy of Turing's work, specifically the ways in which it bridges not only the gap between the sciences and the humanities, but also foundational and practical aspects of science and everyday life. Three sections of the volume are outlined, framing the overarching structure of Turing's intellectual development: (i) Turing on the foundations of mathematics, incompleteness, the limits of analysis; (ii) Turing's Universal Machine, implying the ubiquity of computational processes in our world, exemplified by applications in the early history of voice encryption, the history of computer music, the frontiers of computation, and the topic of emergence; (iii) Turing's work on machines and mind, including his famed "Turing test" as a societal mechanism, the nature of perception as cognition, his views on freedom of the will and the integration of human and machine intelligence, and the developing idea of social algorithms.

This introduction will give an overview of the volume (Sect. 1.1), followed by a brief survey of Turing's life and works (Sect. 1.2), a synopsis of each of the essays in the volume (Sect. 1.3), and a discussion of the most up-to-date volumes, websites and collections of Turing's papers (Sect. 1.4).

1.1 Overview of This Volume

The purpose of this volume is to celebrate philosophical aspects of Alan Mathison Turing's intellectual legacy. While we have not covered every area to which Turing directly contributed—probability, recursion theory, number theory, error analysis, and neuroscience are the most obvious lacunae—we have nevertheless striven to show the profundity and open-ended qualities of Turing's intellectual contributions,

J. Floyd (✉)
Department of Philosophy, Boston University, 745 Commonwealth Avenue,
Boston, MA 02215, USA
e-mail: jfloyd@bu.edu

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_1

both foundational and far-reaching. Although Turing would not have called himself a philosopher—he must be counted primarily as a mathematician—his work has contributed much to philosophy for nearly 50 years, and not only by way of the theory of computability, cognitive science and artificial intelligence (AI).

This is less well appreciated than it should be. Turing profoundly bridged the humanities and the sciences, natural and social, embedding his thought in everyday life as well as in pure philosophy, providing us with powerful and novel ways of seeing and investigating ourselves and our place in nature. His contributions—still stirring lively debate from within the frontiers of science—made him one of the most influential thinkers of all time. Here we aim to characterize how his cast of mind, a versatile and profoundly creative manner of thinking, brought forth a new conception of "foundations" and even of knowledge itself. This cast of mind is what matters to Turing's place within the history and philosophy of science.

To convey the qualities that made Turing's contributions philosophically distinctive, we have arranged the essays in three Parts. Transitions from essay to essay are intended to show philosophical aspects of Turing's thinking, as is the overall arrangement.

Part I, "Logic and Mathematics to Philosophy" collects four essays on foundations, emphasizing Turing's drive for simplicity at the first steps. Cutting through side-thickets of ideological discussion, Turing was able to zero in on an analysis of deductive justification that re-oriented foundations profoundly. Collectively the essays demonstrate how deeply Turing's work is bound up in ancient questions about knowledge, justification, the nature of logic and axiomatization, the finite and the infinite, and the role of language in philosophy.

With the mind of a pure logician and the heart of an inventor, Turing constantly sought to concretize and operationalize his deepest, most general foundational ideas. The essays in Part II, "The Universal Machine: From Music to Morphogenesis", show this side of Turing in action. Focusing on the structure of how we talk about and characterize things (including mathematical objects and processes), he saw how to extract from logic new mathematical structure, allowing us to engineer and offload the work of algorithms to machines. This allowed him to move from fundamental and unresolved questions in logic and probability to concrete computational achievements in encryption and the production of music. He returned at each point to the question of whether there might be wholly new ways of analyzing limitations and complexities, including our own, and so do the essays in this Part of the volume that point beyond Turing's framework.

The essays in Part III, "Human, Machine and Mind" show how at each point Turing engaged himself, with his characteristic penchant for simplicity, in philosophical questions about ourselves and our place in nature. His work on machine "intelligence" opened up profound questions about how we are to conceive of natural processes and evolutions (developmental, social, cultural and ecological), be they human, artifactual, animate, or physical. In the end, he strove for an integrated approach to all of these, characterizing the shifting contributions of computable processes as elements of each.

Collectively these three Parts of the volume show how Turing contributed a new way of conceiving emergent and symbolic phenomena, driven forward by a distinctive way of framing and embedding particular problems and our ways of defining and characterizing them in different forms of thought and representation. Through this embedding emerged—and continue to emerge—new approaches to outstanding scientific, cultural, and philosophical questions.

A philosophical portrait of Turing's distinctive methodology begins to emerge. One may picture him beginning from the hurly-burly of life and science, departing from the usual flow of things toward open, foundational debates at the frontiers, then finding a way to reflect philosophical questions back toward life and nature, rendering them mathematically and then injecting this structuring into ordinary, local experiences of order. At each point, he took words and how we use them to talk about reality to *be* elements of reality, and to tell us how to get at it. He then ingeniously showed how to project the foundations, in an open-ended way, into ongoing experience—where by "experience" he understood something very broad indeed: the collective wisdom of humankind as it has advanced so far, including our everyday ways of classifying, defining, and characterizing processes, things, and procedures.

"Experience" on Turing's view comprises not only patterns instantiated by individual human mentality or neurophysiology, but, in the end, a massive, evolving, cultural setting of humans frequently interacting with one another through speech, *via* and with machines whom they teach, even as the machines teach them to help teach each other by codifying, synthesizing, and re-organizing synoptic views of phenomena. In this evolution, dynamic semantic permeation reaches down into the foundations. For Turing, cooperative and opportunistic integration of innovative ideas with computational power and purpose is what counts, not only subpersonal processing and isolated feats of genius.

One might say, in very broad terms, that Turing revitalized what is best in empiricism for philosophy once it is purged of its rationalistic elements of self-evidence, ultimate verification, and the inner lucidity and transparency of the objects of our consciousness (sense-data, impressions, representations, perceptions, feelings, and/or universals). He rescued the "intuitive" by making rigorous sense of it, but without a theory of rational intuition or truth as such. Instead, he emphasized the nature of searching, and the differing kinds of search that are possible, including vastly cooperative, probabilistic, occasional shot-in-the-dark and partial ones.

1.2 Turing's Life and Works: A Survey

In a biographical memoir of Turing for the Royal Society, Turing's teacher M.H.A. Newman wrote (1955) that

> The central problem with which he [Alan] started and to which he constantly returned, is the extent and the limitations of mechanistic explanations of nature. All his work, except for three papers in pure mathematics (1935b, 1938a, b) grew naturally out of the technical

> problems encountered in these inquiries. His way of tackling the problem was not by philosophical discussion of general principles, but by mathematical proof of certain limited results: in the first instance the impossibility of the too sanguine programme for the complete mechanization of mathematics, and in his final work, the possibility of, at any rate, a partial explanation of the phenomena of organic growth by the 'blind' operation of chemical laws.

In charting the limits and extent of mechanistic explanations, Turing concertedly developed a remarkable sense of where mathematics and logic may and may not structure manageable answers to questions. This required him to have a grasp of where philosophical discussion is going to be inevitable, and where and how limitative results may be parlayed into productive and fruitful approaches and/or used to inspire workarounds.

Already when he was thirteen, he wrote of his work in chemistry that

> … The earthenware retort takes the place of a crucible when the essential product is a gas which is very common at high temperatures. I am making a collection of experiments in the order I mean to do them in. I always seem to want to make things from the thing that is commonest in nature and with the least waste in energy (Hodges (1983/2012), chapter 1).

Turing's biographer Hodges emphasizes that one may already discern here the emerging cusp of what was Turing's intellectual cast of thought, if not his overwhelming passion: a "longing for the simple and ordinary".[1]

Mathematicians generally seek simplicity where it may be found. But what was remarkable in this case was Turing's ability to wrest from tremendously complex, fine-grained thickets of outstanding problems a way of thinking that would set the frame for how to approach things directly, face to face, simply and without the intervention of grand theory or ideology, and yet also rigorously. Like Einstein—also a gifted maker of thought experiments and models—Turing liked to work from scratch, focusing on first principles and thinking things through for himself. This allowed him to confront with fresh eyes, not only issues in the foundations of quantum mechanics, but also in the foundations of logic and mathematics, which were still recovering, during Turing's student days, from the ideologically-tinged controversies between intuitionists and finitists that Einstein had dubbed "the battle of frogs and mice".

Philosophical and speculative questions stretched with constancy through his life.[2] Even before Cambridge, while still in grammar school at Sherbourne, he had developed an interest in quantum mechanics and freedom of the will (Hodges (1983/2012)). He appears to have lost his faith in immaterial accounts of the will by the time he went up to college at Cambridge, reeling from the loss of his friend

[1] Hodges (1983/2012), Chap. 1).

[2] Hodges (1983/2012) *passim*. Later on in his life, one Manchester Philosophy Department event organized by Dorothy Emmet went on past midnight: see Lyn Irvine as quoted in the Foreword to the first edition of Sarah Turing (1959/2012). Moreover, Bertrand Russell, who would nominate Turing (with M.H.A. Newman) to become a Fellow of the Royal Society in the spring of 1951 also sent Turing greetings on the occasion of at least one of his London lectures as a young man. Cf. Sarah Turing (1959/2012), pp. 45,99.

Christopher Morcom.[3] Yet intriguing fragments of lines of thought noted down in a postcard in the final year of his life for his only PhD student, Robin Gandy, state that

> III. The universe is the interior of the light cone of the creation.
> IV. Science is a differential equation. Religion is a boundary condition.[4]

These remarks, even if tongue in cheek, are, ultimately, deeply serious, speculative and philosophical in nature. Showing the influence of Dirac, they suggest a number of different ways of thinking about the limits of scientific explanation. The theme of "the simple and ordinary" is directly connected with limits in the final lines of Turing's last article (1954). Here he suggests that the Gödelian and other impossibility theorems in logic (including his own (1936/7)) serve philosophically in

> ... setting bounds to what we can hope to achieve purely by reasoning. These, and some other results of mathematical logic may be regarded as going some way towards a demonstration, within mathematics itself, of the inadequacy of 'reason' unsupported by common sense.

"Common sense" had of course been a thematic thread in philosophy at Cambridge for some time, as the shackles of metaphysical idealism were thrown off. Turing's approach throughout his life evinced something surprisingly rare: tolerance and plasticity in characterizing and weighing alternative points of view, as well as a remarkable ability to dodge and/or reframe unnecessary ideological controversy.

In 1931 Turing matriculated at Cambridge with a scholarship at King's College. He had already studied von Neumann's axiomatization in *Mathematische Grundlagen der Quantum Mechanik* (1932) before he arrived. In 1933 he gave a paper to the Moral Sciences Club arguing that "a purely logistic view of mathematics is inadequate; mathematical propositions possess a variety of interpretations, of which the logistic is merely one".[5] While this may be taken to reflect a mathematician's insistence that his subject is self-authenticating, unlimited, central to physics, and not bound by any other type of theory—a kind of Hilbertian attitude of autonomy[6]—it also represents entanglement with a distinctive and more purely philosophical point of view.

We do not know whether Turing directly attended any of Wittgenstein's lectures 1930–1936 at Cambridge or saw the dictated typescripts he circulated to students,

---

[3] See his mother S. Turing's memoir (1959/2012) and Hodges (1983/2012), as well as Proudfoot Chap. 12. S. Turing (1959/2012), though she cannot be said to have understood her son, held that Turing was "in limited agreement with Christianity" as a kind of behaviorial device (p. 40), though Hodges (1983/2012) describes him as having lost faith.

[4] The postcard contents are reproduced in Turing and Gandy and Yates (eds.) (1992), pp. 265–9 with helpful commentary; cf. Hodges (1983/2012). See Cooper Chap. 10, Fig. 10.3.

[5] Hodges (1983/2012), p. 88; cf. S. Turing (1959/2012), p. 42; cf. a discussion in Floyd (2012a) and Chap. 5.

[6] Discussions of this idea of autonomy may be found in Detlefsen (1986), Franks (2009), and Sieg (2013). See Floyd (Chap. 4) below for discussion.

but their flavor would have filtered through to him.[7] He certainly had exchanges with those who studied closely with Wittgenstein, and these were sufficiently important to him that later on he held discussions with Wittgenstein himself in 1937 and 1939.[8]

In the autumn of 1933 Turing attended lectures on "The Distribution of Measurements in Scientific Experiments" in which Sir Arthur Eddington discussed the tendency of experimental measurements subject to errors of observation to have an approximately normal or Gaussian distribution. He devised a rigorous proof of this heuristic, demonstrating what we now call the Central Limit Theorem. He was awarded a Fellowship on the basis of this work (1935a) in March 1934 at age 22 (despite the fact that the theorem had, unbeknownst to him, been previously proved in 1922 by the Finnish mathematician Jarl Waldemar Lindeberg).[9] Turing was to remain interested in probability his whole life: it contributed greatly to his wartime work at Bletchley Park[10], and probabilistic, "non-deterministic" generalizations of his later model of computation, the famed "Turing machine" are already mentioned in §2 of his classic (1936/7). Yet Turing's first published paper was in group theory (1935b), improving a result of von Neumann, whom he happened to meet at Cambridge the very month it appeared, and whom he also later encountered at Princeton.

Within 2 years of earning his graduate fellowship, in spring 1935, Turing was brought up to date with recent work on the foundations of mathematics in a course taught by M.H.A. Newman, who would later invite him to a Readership at Manchester to develop computers and eventually pen his obituary for the Royal Society.[11] The course included coverage of Gödel's famed (1931) proof of the incompleteness of arithmetic. This led to what is Turing's most influential and important paper, "On Computable Numbers, with an application to the *Entscheidungsproblem*" (1936/7). The idea apparently came to him 1 day on a walk in Grantchester meadows.[12] It is important to stress the philosophical and foundational importance of this paper, for it forms a kind of base camp for the future trajectory of Turing's work.

Turing (1936/7) resolved in the negative the important (and very general) question of the *Entscheidungsproblem*, or "decision problem" for first-order logic. This was the question whether one can define a decision procedure (a step-by-step, Yes or No, effectively calculable routine) to answer the question whether a particular formula of a formalized axiomatic system is or is not derivable ("decideable") in that logic. The resolution occurs as an "application" of the theory his paper advanced, appearing in its final section. Turing had not only resolved a pivotal problem. By projecting it into a cluster of a whole range of other problems, he had constructed a

[7] Hodges (1983/2012); cf. Floyd Chap. 5.

[8] Wittgenstein (1979), cf. Floyd Chap. 5.

[9] Zabell (1995), (2013), cf. Hodges (1983/2012), p. 88.

[10] Good (1979, 2000a, b); cf. Hodges (2003).

[11] Newman (1955), cf. Grattan-Guinness (2013).

[12] Hodges (1983/2012), pp. 96–100.

whole new way of thinking. The notoriety he ultimately won for this achievement was thus not merely one of achieving an isolated result, famed though it was at the time.

To appreciate the significance of Turing's (1936/7), we may review its backdrop in the Hilbert school. Having developed an axiomatic, formalized approach to gauging the relative strengths and structures of mathematical theories, this research strand focused around Hilbert aimed to secure a foundation for mathematics by obtaining proofs of consistency of its theories. The project was to scrutinize "metamathematically" the behavior of the formalized axiomatic systems alone, invoking neither infinite objects and processes nor results about them. In this way philosophical questions about the foundations of mathematics could be addressed directly by mathematics itself. The idea—initially plausible—was that a "finitary" approach, relying on finite sequences of manipulation and substitution of formulas and terms by ordinary "calculable means", could establish the consistency of analysis, and ultimately even arithmetic itself.

There were a triad of beliefs here. First, Hilbert believed that every properly formulated mathematical proposition is either provable or refutable, i.e., mathematics is *complete*. Second, Hilbert aimed to prove the *consistency* of mathematical theories by showing that a proof of contradiction cannot possibly arise in the formal systematization of the theory. Third, there was the *Entscheidungsproblem*: find a general method to determine, Yes or No, in a finite number of steps, whether or not a given sentence of a first-order formalized theory of mathematics does or does not follow from the axioms.[13]

Gödel's (1931) proof of the incompleteness theorems for arithmetic forced a modification of the first two legs of this triad of beliefs.[14] It showed, first, that any formal theory whose axioms and rules of derivation may be recognized in a finitary manner and which contains some arithmetical axioms for induction will be, if it is consistent, *incomplete*: that is, it will contain formulas P such that neither P nor its negation $\sim P$ can be (finitistically) derived from the axioms, so that each might be added to the theory consistently. Second, since this proof may *itself* be formalized within any such theory (being finistic itself), Gödel showed that no such theory can finitistically prove its own consistency, assuming that it is consistent. The Hilbertian hope of founding consistency proofs of arithmetic and analysis directly on "finitary" processes was thereby doomed.[15] What remained at this point most pressing

[13] Gödel's (1930) proof of the completeness theorem of first-order logic showed that a sentence is valid if and only if it can be deduced from the axioms of first-order logic, so that the *Entscheidungsproblem* may be regarded as asking for a "definite method" to decide whether a given statement is provable from the axioms using the rules of logic alone, i.e., whether it is valid in every structure satisfying the axioms of that system.

[14] Franzén (2005) contains a crisp exposition of the theorem, discussing at length what it does and does not show.

[15] Of course partial consistency results of fragments of arithmetic might still be sought, and so they were, among others by Turing.

was the issue of the *Entscheidungsproblem*. To resolve it, one had to make *general* sense of the notion of an "undecideable" proposition.

A negative verdict on the *Entscheidungsproblem* was proved by Alonzo Church (1936) in a paper finished just before Turing completed his own. Proceeding unaware of Church's work, Turing offered a very different kind of proof (1936/7). His method was based on an especially interesting analysis of the notion of a "computable number" as one calculable by a "machine". The paper—which Newman initially greeted with disbelief, given its apparent naïveté[16]—turned out to hold the conceptual key to clarifying the foundational issues at stake in a way that Church's did not.

For Turing analyzed the notion of "computation" in terms of an analysis of *human* activities of computation according to a rule, given infinite paper. The developed machine-model was then superimposed on a theory of calculative manipulations of strings of characters. Turing also showed the existence of a universal machine, one that could do the work of all other computing machines: order all specifications of particular Turing machines lexicographically, and embed their specifications into a single machine (since there is an unlimited tape, this adding on is always possible). Turing's universal machine nicely closes off the domain of algorithms, for the class of functions it uniformly defines and computes cannot be "diagonalized" out of by way of a self-referential or looping argument: partially defined functions block this move.

The universal machine served a theoretical aim crucial for understanding, not only the Hilbert program, but also the scope and character of Gödel's proof of incompleteness. Gödel knew that the generality of his proof could not be fully gauged until the applicability of his coding devices to *all* formal axiomatizations was demonstrated; his (1931) proof had of course relied upon a standard but single such formal system. What was required therefore was an analysis of what a formal system, in general, in the relevant sense, *is*. This, Gödel came to agree, was provided by Turing's (1936/7).[17] Much later it would be seen that Turing had provided the blueprint for the idea of a stored program computer: one that works on its own commands.[18]

Recognizing that Turing's (1936/7) approach was distinctive, Newman urged its publication. Turing added an appendix proving that his conceptual device—which Church would label, in his review of the paper, a "Turing machine" (Church (1937))—was able to calculate all and only the intuitively computable functions that had so far been defined by Church and his student Kleene at Princeton in terms of their "λ-calculus", a system for gauging the definability of effectively calculable functions.

This injected Turing's work into the heart of then-nascent computability theory, bringing to the fore the notion of a "mechanical process". Such a process Church

[16] Cf. Hodges (1983/2012).

[17] See the Postscriptum (1964) to Gödel (1934), Gödel (1967) and compare Kennedy Chap. 3 for further discussion.

[18] Cf., e.g., Davis (1987).

emphasized as something very general, tied to the notion of a "computing machine". Church took this to be, not merely motivated by the intuitive idea of a human computer, but, more generally, a kind of thing.[19] Much of the controversy today surrounding the concept of "computation" surrounds that generalization and the justification, if any, for it.[20]

For Turing's journey from the heart of philosophical questions about the foundations of mathematics to the notion of "computation" was unique, setting him apart not only from Church, but also from Gödel.

By 1934, building on work of Herbrand, Gödel had developed and lectured at Princeton on an expanded version of a notion broached in his incompleteness proof, that of a "general recursive function", one using limited means of definition to ensure "finitary" effectiveness. Kleene developed the idea, bringing forward an equational system of equations nowadays known as "Herbrand-Gödel-Kleene" or "HGK" systems. *Church's Thesis*, first framed in 1934, is the claim that the intuitively "effectively calculable" functions are just those that are λ-definable; the thesis was soon amended to concern those functions that are HGK definable: these classes are coextensive, as was quickly shown by Church and Kleene.[21]

Yet Gödel was not satisfied by Church's "thesis" at the time. Why should *these* systems of equations and the λ-calculus—difficult to work with and survey—be regarded as telling us what "effectively calculable" and "formal system" *mean*? Not until Turing's (1936/7) paper was this made clear in Gödel's and others' eyes. Turing himself proved that the class of Turing-computable functions is identical with those of the λ-definable and HGK definable ones, completing the reduction (1937a).

"Church's thesis" continues to be widely debated nowadays, precisified in a variety of ways. Some hold that it cannot possibly be proven (since it equates an intuitive notion ("effectively computable" or "effectively calculable in a logic") with a rigorous mathematical one, and no such equation is provable. Others argue that it *can* be proved, if we treat the concept of "computable" with sufficient care, either as an evolving, open-textured kind of notion or as a refined logical conception. Others deny that the "thesis" counts as any kind of "hypothesis" at all. Most would sharply distinguish the "physical Church-Turing thesis"—applied to arbitrary physically embodied machines—from the "Turing thesis", which is restricted to the (idealized) arena of a human computer working according to a rule.[22] The reader will find much comprehensive discussion elsewhere, both of the thesis itself[23] and the more general historical question of what the influence of Turing really was on the history of computing.[24]

---

[19] Hodges (2013).

[20] On this see Davis (1982), Sieg (2009), Mundici and Sieg Chap. 2 and Kennedy Chap. 3.

[21] See Kleene (1981a, b) discussed in Kennedy Chap. 3.

[22] See Gandy (1988), Duwell Chap. 8.

[23] Herken (1988), Olszewski, Wolenski and Janusz (eds.) (2008), Copeland, Posy and Shagrir (eds.) (2013); cf. Copeland (2009) for an introduction.

[24] Controversy has arisen over Turing's role in stimulating von Neumann, who authored in late June 1944 the "First Draft of a Report on the EDVAC" (which defined the critical "stored program"

The success of Turing's (1936/7) led to his invitation to Princeton as a Procter Fellow in 1936. He worked for 2 academic years to solidify his research. He constructed algorithms for computing real numbers (Becher et al. 2007). A paper on fixed point combinators in Church's λ-calculus (a main ingredient for writing recursive programs in functional style) appeared (1937b). In his dissertation, written under Church ((1938c/2012), cf. (1939)), Turing began charting how much of mathematics might be captured by eliminating "as far as possible the effects of Gödel's theorem" (1938c/2012 §9). The idea was to study definability in "ordinal logics" to "get around" incompleteness: Gödel's (1931) had really given a constructive procedure for generating stronger and stronger logical theories by adding recurrently undecideable propositions.

Turing postulated a series of "oracles", specific (infinitary) knowledge claims of the form "at some point Turing machine*M* prints *x*", despite the fact that no Turing machine is able to (computably) determine whether these are true or false.[25] This opens up the whole field of "relative computability".[26] For example, applying his (1936/7), Turing was able to show that while there is an "oracle" or *o*-machine that can solve all number theoretic problems, the problem of determining whether an arbitrarily selected description number for an *o*-machine is indeed the number of a machine that will print an infinite number of digits as required is not a problem that any *o*-machine can solve.

Turing waxed philosophical about this. His heuristic picture is that one may enlarge the power of problem-solving by allowing for (partial, controlled, fallible) uses of "intuition", and then, by examining our descriptions of this, chart the limits of incompleteness in a theoretically systematic way. In §11 of (1938c/2012) he "rather schematically" takes mathematicians to be exercising three different "faculties": *judgment*, *intuition*, and *ingenuity*. Judgment, "that most important faculty which distinguishes topics of interest from others", he does not analyze. "Intuition" he took to consist "in making spontaneous judgments which are not the result of conscious trains of reasoning", judgments that "are often but by no means invariably correct (leaving aside the question as to what is meant by 'correct')". This development of an idea of defeasible, non-inferentially grasped belief (cf. Russell (1912/1990), XI) is supported by "ingenuity", which "aid[s] intuition through suitable arrangements of propositions, and perhaps geometrical figures or drawings. It is intended that when these are really well arranged the validity of the intuitive steps which are required cannot seriously be doubted".

Turing characteristically avoided issues about the nature of truth, remaining descriptive and procedural. He pointed out that detailed mathematical argument may verify the correctness of an intuitive judgment, but that any such "will also involve intuitive judgments, but they will be ones less open to criticism than the

concept), and in aiding the development of the "Manchester Baby". See Davis (2000/2011), Copeland (2006) and Copeland (2011a, 2011b), 2012) as well as Dyson (2012).

[25] A useful explanation of the significance of the dissertation is given by Feferman (1988) who developed Turing's work (1962). Further relevant commentary and correspondence is to be found in in Copeland (2004), pp. 125–204 and Cooper and van Leeuven (eds.) (2013), pp. 145–210.

[26] Cf. Mundici and Sieg, Chap. 2.

original judgment". He thus drew away from eliminating ultimate forms of "intuition". "Ingenuity" may be replaced to a large degree by "patience", the reliance on "intuition" minimized through the use of a formal logic. But, because of the limitative results, "intuition" would never be entirely eliminated.

As we see, Turing was not only occupied with computational applications, but with their theoretical limitations and the practical and philosophical impact of these. If the Second World War had not intervened, his research path might have been different.[27] But he made do with whatever was ready to hand.

The line between his pure and applied mathematical work was fluid. He authored two papers on approximable groups and extensions of groups (1938a, b), later applying his concept of a stored program computer that could "switch from calculating the energy levels of the neon atom to the enumeration of groups of order 720" without any augmentation of its internal workings, but only "paper work" by programmers—a great advantage over the analog devices used up to that point (1947). Machines could have mathematics offloaded onto them, and, embodying logical operations, the whole system of humans and machines would become more "intelligent". In 1939 Turing framed a theoretical method for calculating the Riemann zeta-function (1943, 1943/1944); he later used the Manchester University computer to investigate its zeros (1953a). A paper on numerical analysis applied to matrices gave both absolute and statistical bounds for rounding off errors, a useful approach for future computational work (1948a/1992).[28] An unpublished paper on permutation groups (Turing (?)) was motivated by his study of the German enciphering "Enigma" machine, in the course of breaking encoded messages.

In the public's eyes Turing is perhaps most widely remembered for his contributions to cryptography, specifically, to breaking the German "Enigma" code at Bletchley Park.[29] But he is also increasingly remembered, more generally, for sparking with his universal machine the idea of a modern stored program computer, as well as working persistently to realize the concept with minimum sorts of hardware and an emphasis on human programming.[30]

Definability and "types" (kinds, classifications) remained a constant concern in both practical and foundational directions throughout the war. Turing developed *in situ* a strong sense of the importance of level "between" foundation and application, the difficulty of putting the program that moved the machine into higher-level, synoptic forms so that the humans and machines could verify, improve, and utilize them. He published and wrote papers on λ-definability and Church's system (1941, 1942), Turing and Newman (1942)), moving on to a series of logical papers on the theme of "types", i.e., categorical treatments of higher level reasoning (1943/4, 1944/5, 1948b/1992). Gandy reports that Turing "set some store" by his idea of

[27] Hodges (1983/2012), (2013).

[28] See Blum (2013) on this.

[29] Turing (c.1940), Hodges (1983/2012), Kahn (1991, 1996), Hinsley and Stripp (eds.) (1993), Copeland (2006).

[30] Davis (2000/2011), Copeland (2011a), (2011b), (2012); cf. Davis's Chap. 5, Copeland and Long Chap. 8.

"equivalence of two logical systems" and intended his "concealed type theory" "to be the beginning of a bridge between type theory and set theory".[31] This amounted to a way of presenting forms of argument that would greatly simplify, and in the end even eliminate the need for mathematicians to master the thickets of symbolism evinced in, e.g., *Principia Mathematica.*

After the war Turing designed the "Automatic Computing Engine" (ACE) during his time at the National Physical Laboratory, Teddington, London, 1945–1947 (Turing (1945/1992)). He stimulated the ACE's development and played a large role in the very first pilot realization of the stored-program concept at Manchester University. Newman hired him as a Reader in 1948 and he remained there until his death in 1954 (cf. Turing (1946, 1947/1992,1948c/1992).[32] The Small-Scale Experiment Machine (SSEM) project at Victoria College, known as the "Manchester Baby", was run in mid-June 1948 with important input from Turing.[33] As more becomes known about Turing's efforts as a programmer, his facility as a software engineer becomes clearer (cf. Turing (195?)). His method for annotating programs to check them for correctness has been said to have presaged some of the earliest approaches that were only developed 17 years later.[34] He predicted with remarkable prescience the importance of such notions as that of a subroutine, computer-based theorem proving in software development, and machine networks.

He still kept his hand in with pure mathematics (1950a), but also made pioneering efforts in founding the field of AI and contributing to debates about the nature of "intelligent machinery" (1948c/1992,1950b).[35] He not only investigated the application of digital computers to games such as chess (1953b), he also lectured to learned societies, magazines and radio audiences to explain the importance of his research (1949, 1950b, 1951a, 1951b). In (1950b), the best known source for his "Turing Test", he introduced the relevant protocol in an "imitation" game in which an "interrogator" was to determine, by asking questions through a keyboard alone, behind a screen, which of A or B—the two dichotomously sexed humans devising answers to the interrogator's questions—was in fact female, and which in fact male. This "sorting" task establishes a baseline (statistical) comparison for the next situation, in which a machine takes the place of one of A or B, leaving the interrogator to determine which is machine, and which human. Presumably, at this point the sex of B does not matter, male or female (although there has been some discussion of whether B was to have been male or female).[36]

[31] Turing, Gandy and Yates (eds.) (2001), pp. 179, 266.

[32] Discussed in Copeland (2011a), (2011b), (2012).

[33] See, for discussion, Copeland (2004, 2006, 2011a, b, 2012) and Copeland and Long Chap. 8.

[34] Turing (1949), cf. Jones (2013).

[35] See Winston's Chap. 10 for a discussion.

[36] The Introduction to Turing's (1950b) in Copeland (2004) pp. 433–440 shares some of the history of earlier framings of the test that are due to Turing. More on the parameters of the Turing Test may be gleaned from papers in Moor (2003) and Shieber (2004), as well as the commentaries on Turing (1950b) in Cooper and van Leeuven (eds.) (2013), pp. 551–622. Cf. Oppy and Dowe (2011) for an overview.

Turing's late writings on morphogenesis and phyllotaxis—the mathematical and chemical basis of the evolution of "emergent" forms in nature—were also produced during this time (1952).[37] These harbingers of future research in computational biology and "artificial life" picturesquely suggest a connection between "higher order" computational types and the concept of "emergence". It is clear that Turing had in mind an ambitious focus on computational aspects of science and neurobiology at the time of his death in 1954.

Much of Turing's wartime work remained quite unknown for a long time due to the Government Secrets Act in Britain; two of his reports on the applications of probability to cryptography from 1941 were only declassified in 2012[38], and, presumably, there is more to come. His life was lived in the shadows of intelligence and the use of the label "homosexual" to discredit and manipulate would-be spies, and his final post-War years, when he travelled to Germany to investigate the state of science there, remain still relatively unresearched.[39]

Turing's final article (1954) gave a sophisticated popular treatment of the concept of a mathematical puzzle, returning to themes treated in his earliest logical work. The paper is deceptively simple: an apparently easy, do-it-yourself piece, it is in fact deeply connected to the closely related foundational work of Emil Post, another pioneer in the foundations of logic and mathematics with a penchant and gift for simplification, who happens to have offered a precisely coincident analysis of the foundations of logic in his (1936), just at the time Turing's (1936/7) was completed.[40] The "confluence" of their models of computation, based on the intuitive idea of a human working step-by-step in accordance with a rule, is one of the watersheds of twentieth century science and philosophy, one that made possible our era of computation.

1.3 Précis of Essays in This Volume

1.3.1 Logic and Mathematics to Philosophy

In Part I of this volume Mundici and Sieg (Chap. 2) and Kennedy (Chap. 3) offer insightful accounts of the mathematico-logico-philosophical backdrop to Turing's analysis in his (1936/7). Together they show why it is, ultimately, his most fundamental scientific paper. They synthesize and appraise a great deal of historical material, leading the reader through the most significant historical and mathematical literature summarizing the "confluence" of ideas in 1936. Ultimately, however, they

[37] Cf. Turing and Saunders (1992).

[38] According to Beebe's bibliography (March 15, 2015), discussed below in Sect. 1.4.

[39] Hodges (1983/2012), note 8.46; cf. Sommaruga and Strahm (2015).

[40] Davis and Sieg (2015).

adopt differing philosophical emphases on the nature and significance of the "confluence".

Taking the part of Hilbert, Mundici and Sieg present a way of understanding Turing and Post that stresses the *axiomatizable* rigor of their analyses. This shows how Turing's work, while intuitively accessible and vivid, meets at the same time the most stringent epistemological standards for mathematics itself. From Mundici and Sieg's point of view, the "confluence" was *not* historically or mathematically accidental, but emerged through precisely the right conceptual focus on clarifying what the Hilbert school had regarded as essential elements of the "finitary" point of view. This was needed to implement the axiomatic method mathematically, and, subsequently, computationally.

The key move is the reduction to a theory of human activity on strings of symbols. Turing remained, until his death, somewhat guarded about the "enigmatic" or "intuitive" status of his (1936/7) conceptual clarification. Yet, Mundici and Sieg argue, the axiomatic adequacy of the theory can *itself* be shown both conceptually and mathematically, *via* specific axioms for a theory of "strings" involving conditions of boundedness and locality and thereby computationally. These conditions may themselves be incorporated into an abstract, axiomatic characterization of a *computable process* and then computationally implemented.

Stressing how Turing's analysis leads us directly into problems of real vs. polynomial time, Mundici and Sieg argue that "the step from decidability to tractability is the birth of complexity theory". After characterizing NP-completeness and enumerate several key reductions of problems to NP-complete ones, they close with an allusion to Hilbert's famous (1900) lecture on the most important outstanding mathematical problems of the last century, bringing the point of view up to our time: the P/NP problem, they say, "may be considered as a sort of *Entscheidungsproblem* of our century".

Kennedy takes our eyes off of strings as manipulated by human computors—the theory sitting at the foundation of the Hilbertian approach of Mundici and Sieg—and lifts our eyes up to the heights of the infinite, aiming to grapple conceptually and philosophically with the ideas of formalization and axiomatization as such. Turning toward the interplay of Turing's work with that of Gödel—who after all initially framed the P/NP problem, in a letter to von Neumann (Gödel (1956))—Kennedy returns us to pure foundations of mathematics, encompassing not only the development of computability, but also the theory of definability and provability as they were developed by Gödel after 1931.

Inspired by Turing's "intuitive" analysis of "formal system", the Gödelian perspective Kennedy defends is that of "formalism freeness", the ideal of a ground for the possibility of sharp mathematical analysis that allows for, by explaining, the epistemological relevance of the "confluence" of certain analyses *without* relying on a particular formalization. She emphasizes the purely philosophical resonances of this research, insisting on our facing the fact—as she argues mathematicians always do, practically speaking—that there is a "bright abyss" between mathematical intuition and mathematical fact, or the "raw" and the "cooked": a place of, so to speak, *faith*. Inspired by Gödel's praise of Turing's analysis in his (1946) Princeton

Bicentennial lecture, she argues that this perspective is an ongoing project, especially for the notions of "provability" and "definability".

Can a robust understanding of these notions be found that seems as *right* as Turing's analysis of computability was, and is "formalism free"? Will an absolute notion of "definability" be framed? The path to this, she thinks, will be longer and more involved than it was for Turing in 1936. As she explains, absolute or grounding clarifications of provability and definability are continually being sought by the additions of axioms of infinity by set theorists. A class of "hereditarily ordinal definable sets" ("HOD") has emerged, those sets whose transitive closure is itself ordinal definable. But, she argues, the perspective stressed by Gödel and Turing remains. The search for "logical autonomy" requires us to look "in two directions at once", i.e., toward mathematics done purely, and within the "pleasant and comfortable vagueness of natural language", and also at the same time toward precisification. As a result there must remain a "bright abyss" of faith, she argues, despite the search for absoluteness. While sophisticated parametric treatments of logics may and must occur, we always face the difficulty of deviant encodings. In terms of a Turing machine, inputs and outputs must be encoded in finite strings in order for the model of "effectivity" to work. This in turn, she argues, will introduce circularities again at the bottom, for there is no general method for sorting computable from uncomputable encodings: all we have is a list of "what works".

The next two essays move from this perspective on meaning to ask how we are to understand Turing's procedures *extramathematically*, i.e., more purely philosophically. Both Gödel and Turing were convinced that this step would be necessary to confront the foundational level.

In Chap. 4 Parikh and Renero use Turing's analysis of computation to criticize the ancient tradition of analyzing knowledge in terms of "justified true belief", the so-called "JTB" theory. This is the claim that a subject S knows that *p* if and only if

1. *p* is true
2. *p* is believed by S, and
3. S is justified in believing *p*.

Plato's *Theaetetus* famously begins and ends with difficulties surrounding the notion that knowledge is perception. In considering whether knowledge is true belief with an account (*logos*) or justification, Socrates analogizes justification to the analysis of the first syllable of his own name "SO" into two letters "S" and "O" ((1997), *Theaetetus* 202ff). The difficulty is that knowledge of the relation of these parts appears to rest upon knowledge of the whole, and vice versa, the letters themselves correlated with different sounds and syllables. In the end, even to form a belief one would seem to need, not merely perception, but *appropriate* perception.

"Gettier cases" (1963) demonstrate that the above three conditions are not jointly sufficient for knowledge in any case. The contents of perception and knowledge, when articulated in justifications in language, evince elements of variability depending on judger, audience, external environment, relevance of alternatives and other features of the occasion of judgment. Parikh and Renero report that in ancient and medieval Indian epistemology one already finds the point made. But they criticize

the JTB tradition from a different angle, focusing on how the notions of *analysis* and *justification* are handled by Plato, Wittgenstein and Turing. They argue that while in certain cases knowledge may well be analyzed as perception, in other cases we must attend to the fact that "there is no *one size fits all* justification".

Their account begins from the assumption that there is only partial knowledge held by each member of a knowledge community at any time. From here they show that even if from outside the community a total-knowledge rule of justification may be specified, it may in fact never be used, different justifications being appropriate for different agents depending upon their situation. Knowledge and justification still work, and can have their logic represented, but only if norms of accessibility and appropriateness are brought into our representations of justifications of knowledge. Since multi-agent systems are the norm, rather than a simple add-on to the individual case, the manifestation of knowledge in linguistic articulation should not require, as the JTB appears to, any such notions as community- and/or operationally-transcendent "content". Instead, while there is in the end inevitably a dependence on how individual participants are able to regard that content, and proceed with it in a public setting, understandings are themselves constructed, by us. The inevitable residue of "perception" (as Plato calls it in the *Theaetaetus*) directs us toward the very foundations of logic itself.

In Chap. 5 Floyd shows how central the philosophy of logic was for Turing throughout his life, arguing that it was more fundamental to his work than was philosophy of mind. She holds that Wittgenstein and Turing shared a distinctive approach, in which everyday "typings" of words are taken seriously as *loci* of logical analysis, and in which the notion of "logic" itself is generalized to include all of human regularity, procedure, and systematization. Whereas Parikh and Renero argue that perception (understanding) is necessary to analyze knowledge properly, Floyd argues that an "anthropological", comparative method to the foundations of logic is what lent Turing's analysis of computability its vividness and "rightness" as an analysis.

As she explains, Turing's Cambridge *milieu* deeply shaped his philosophical sensibility, which was in turn crucial for his foundational work, not only "On Computable Numbers" (1936/7), but also his dissertation (1938c/2012, 1939) and two later papers, including (1944/5) (where he explicitly acknowledged Wittgenstein for his way of framing the theory of types) and his last paper (1954). At Cambridge questions about the *nature* of logic and of "common sense", as well as the relations between mathematics and logic, were addressed as central ones over several decades, and by philosophers and mathematicians alike—something of which Turing was fully aware. Turing's exchanges with Wittgenstein and those in his circle, as well as Wittgenstein's debates with Russell, Hardy, and Littlewood all reflect these traditions, which contributed crucially to Turing's intellectual development.

On this reading, the Turing machine is less a general purpose reasoning device than a widely-applicable parameter for titrating routines so that we can discuss their meanings and significance. If the "space of reasons" is not *one* space, but instead one drenched with human interaction, requiring a variety of agreements in judgments and levels of conceptual articulation—"common sense"—then our very

notions of logic and reason-giving, and indeed of "ontology", become normative, contingent, and opportunistic, and our own judgments contingent upon understandings with others. The outcome of Floyd's argument is an anti-reductionist reading of Turing, one that shows his fundamental importance for all of philosophy, and not only philosophy of logic and mathematics.

1.3.2 The Universal Machine: From Music to Morphogenesis

The essays in Part II of the present volume explore the nature and implications of Turing's concept of a universal machine. The conceptual breakthrough is characterized, and then its applications. In turning to the latter, several questions are explored in detail. Are the portions of nature computationally modeled and represented in (computer-generated) images to be regarded as metaphors, explanations, or instantiations? Are they all four? How? Is Turing computability just a model-generating perspective, or is it descriptive? Is it explanatory? And at how fundamental a level? Can it replace or explain causality if we take the point of view that information is not "flat", but "typed"? And how has the model affected the history of computation itself?

Davis's Chap. 6 focuses on the *ubiquity* of computational processes flowing from Turing's universal machine. He argues that the fact that the ancient concept of *algorithm* is now understood in full generality has led to a paradigm shift, not only in our notions of computation and intelligence, but also of evolutionary biology. The crucial issue opened up by the universal machine is the concept of a stored program computer. While the Turing machine is *initially* envisioned with mechanical parts, and the code on its tape is conceived as a program or command to act, nevertheless in its step-by-step actions, the code *itself* is just more data to be worked on. So a Turing machine is not only a particular device instantiating a program, but also, through the universal machine, an interpreter and modifier of programs and outputs, and in fact even its own hardware, thus in principle "doing the work of all". This, Davis argues, secures the *fluidity* amongst categories that have often been conceptually separated. The usual supposition was that in dealing with a calculating machine three categories had to be kept distinct: *machine* (hardware), *program* (software), and *data*. Turing's analysis shows the limitations of this distinction, and in fact the principled impossibility of holding it across the board. Hence the ubiquity of computation.

Of course, a crucial idealizing assumption in Turing's model is the *unlimited* availability of tape, with no time constraints on finding, accessing, and implementing the directions embodied in codes. Turing's appreciation of the delicate interplay between infinitary and finitary aspects of reasoning, as well as his knowledge of physics, was crucial to his practical instantiations of the idea of the universal machine. In fact any particular physical device is at best an *approximation* to the universal machine, as can be illustrated by punching into a handheld calculator a sum too large for it to handle. Davis points out that Turing himself was well aware

of this point, stressing the importance of quickness of access to memory and minimal code software in the actual design of machines.

Davis concludes with two philosophical proposals. First, he boils down the dispute over "artificial intelligence" into two basic theses necessary and sufficient for accepting that concept: (1) The human mind is a function of the brain and (2) The brain is a computer. Adopting a task-oriented approach, Davis predicts that as Turing's "prophetic vision" becomes further and further realized by devices with quicker and quicker random access memory, better and better accessible data, and more carefully targeted software, people "will not hesitate to use the word 'intelligence'" in referring to the behavior of the objects (the usage is already here, with increasingly everyday devices such as "smart" phones).

Second, Davis revisits evolutionary theory inspired by the importance of random access to codes as part of the universal machine model. Long stretches of DNA that were once dismissed as "junk" because they do not code specific proteins are now the focus of biological investigation as they are preserved over long periods of time. Perhaps, Davis conjectures, these function as "speed up" elements, helping to retain modification of the germ-plasm's access to DNA "information" in ways "fitted" to evolutionary survival. If so, evolutionary change would not have been entirely based on "random" variations, but would include what Davis calls a "quasi-teleological" aspect, a bias toward the evolution of "speeded up" processing of computational activities.

In Chap. 7 Bauer shares recent discoveries he has made about Turing's place in the early history of voice encryption, also known as "ciphony". Speech is very difficult to encrypt, in comparison to text: it has many redundancies in it for ease of understanding in conversation; thus it is that much more challenging to encode it in an apparently random manner. Bauer surveys the earliest encrypters from the 1920s, in which high tones and low tones are swapped; the weaknesses of this are apparent: tones near the middle are nearly unchanged and anyone who can build an inverter can easily recover the original. By 1937 AT&T and RCA had developed the A-3 scrambler, which in addition shuffled or transposed the subbands; but it still was not very secure.

Nevertheless, Churchill and Roosevelt communicated by means of the A-3, and by September 1941 the Nazi regime was listening in on their conversations. In fact all parties were aware of the limitations of the A-3 scrambler, and expected that it could be easily deciphered. General Marshall conveyed his warning about the Pearl Harbor attack in text, not trusting the A-3, and his warning was only received after the attack had begun. The Allies soon turned to developing a better mode of ciphony, and, extending existing "vocoders", which rework digitized speech for the purpose of data compression. They developed SIGSALY, a system that had 7 different secret names among allied workers: a measure of its importance. SIGSALY had an additional encryption process involving a separate and independent random key for each channel: each key was kept on a 16 inch record, and was to be used once, and then destroyed. A backup key was required, and SIGBUSE, a large pseudorandom key generator, was devoted to this purpose.

Turing was brought to the United States in January 1943 to evaluate SIGSALY, spending time at Bell Labs and meeting Claude Shannon. On his way back to England in March 1943, Turing felt inspired enough to work on another ciphony machine, "Delilah". His report on this machine was published for the first time by Bauer in the magazine *Cryptologia* in October 2012. Turing's distinctive approach to the problem was to get rid of the extra back-up key (SIGBUSE) and make sole use of a keying system. He "rid himself of everything that wasn't absolutely essential", ditching the vocoder component and attempting to encipher the analog voice signal directly—a signature of "doing with less" that we have seen Turing engaging in before.

SIGSALY was never broken: the Germans never even realized that it represented speech. When it was finally declassified in 1976, a large number of patents, applied for decades earlier, were finally granted. Bauer points out that although it is technically more difficult to render speech into difficult-to-decipher code, text systems take longer to encipher and decipher than voice systems. Instant voice communications make a tremendous difference in the heat of battle, as the United States' reliance on Navajo code talkers showed in the Second World War. Voice encryption remains of crucial importance today, as humans are replaced by machines.

In Chap. 8 Copeland and Long pursue the wide-ranging application of Turing's ideas to everyday life, taking up a fascinating journey into the history of computer-generated music—one that employs digital "archaeology", i.e., digitization by the use of stored-program computing—to explore the history of Turing's singular contributions to software engineering. Although it is "an urban myth of the music world" that the first computer-generated musical notes were heard in 1957 at Bell Labs in the USA, Copeland and Long show that 9 years earlier Turing's Computing Machine Laboratory at Manchester had already brought computer-generated melodies to the human ear, in fact broadcasting the results widely.

First, Copeland and Long set the stage for their analysis by reviewing the most important cast of characters in the Manchester Laboratory, explaining how the idea of a "hooter" to generate musical notes was conceived. They document how the programmers augmented Turing's note-playing subroutines to create computer-generated melodies, offering a detailed look at these very early exercises in software development, showing step-by-step how the melodies were produced. They review in detail and clarify Turing's own programming manual for the Manchester machine—the first ever written for a stored-program computer Turing (195?).

Second, by analyzing Turing's notation for the machine's instructions, they are able to show *how* computer generated melodies were produced through transformations of the code. Since these sounds were recorded by the BBC in 1951, Copeland and Long are able to computationally analyze the actual recordings, reverse-engineering the algorithms that were actually devised. This form of "digital archaeology" sets a high standard for future historical efforts directed at the digital era itself, showing how "digital humanities" will become a necessity for future research into history. It also shows the enormous potential for Turing's model of computation to help guide data-driven research in nearly every discipline. Without Turing's universal machine and his specific emphasis on the importance of software engineering

and types at the level of the programmer end, Big Data would be impossible to create and utilize.

The next two chapters in the volume consider the limitations of Turing's analysis of computability, probing its character as a scientific paradigm of explanation.

In Chap. 9 Duwell argues that Turing failed to give us a general analysis of the notion of *computation*. In light of the axiomatization of Turing's model discussed by Mundici and Sieg (Chap. 2) in which boundedness and locality assumptions are made explicit, Turing's conceptual analysis would, it seems, be invulnerable to a critique from the side of metaphysics. Yet, Duwell argues, an analysis of computation should tell us "what it takes for a system to be a computing system". This imports a significant explanatory demand: to identify *essential explanatory and causal* features of computation (whereas Turing was, in the first instance, offering a conceptual reduction of prior ideas to his own). Duwell is demanding that an account explain to us why and that a rock is not actually a computing system, or computing. We lack, he thinks, an agreed upon set of assumptions about what makes a computer a computer.

First Duwell rejects what he calls the "simple mapping" view of computation that exploits Turing machines directly, by regarding a physical system as computing in virtue of sequentially entering a series of different states (ordinarily equivalence classes of microstates) that a Turing machine goes through when computing. He enumerates a series of difficulties with this view, including the so-called Putnam-Searle objection: any 1-1 mapping account fails to be explanatory, because so many states are physically accessible that states of macroscopic systems are easily found to match with states of Turing machines.[41] Modeling is not explaining. On the simple mapping view, almost everything turns out to be a computer, thereby trivializing the notion of computation.

Duwell's ultimate claim is that one cannot think of "steps of computation" as one would a string of digits. One reason is that we may try to find machines that "take fewer steps" than Turing machines to perform computations. Duwell's idea of "natural computing" is that of specifying a computational object in spatiotemporal terms, defining components by their function and organization insofar as they "process vehicles according to rules".

Second, relying on Piccinini's mechanistic view, Duwell sets out the notion of a "measurement based quantum computer", a generalization of a circuit computer. Given quantum effects, patterns can be performed simultaneously and initially in the computation, realizing rules but not depending on results of earlier measurements. The behavior of "qubits", although they are the results of random measurements, "are nonetheless exquisitely correlated with one another". Useful computational results might then occur insofar as they are generated by quantum entanglement, harnessing its power.

In the end this argument appears to take Turing's analysis of "step" for granted as a parameter, for any mathematical articulation of "natural computing" would devolve upon some notion of proof. But Duwell holds that we need to be "open to

[41] See Putnam (1988), Searle (1992).

the possibility that computers might be able to do things in ways that humans cannot", and that there may be "computers that solve problems very differently than Turing machines"—including, perhaps, the human brain itself.

In Chap. 10 Cooper widens the perspective on limitations facing the "classical" Turing paradigm, emphasizing "types" and "definability" as central notions, hence incompleteness. As a mathematical "recursion theorist" in a field hugely indebted to Turing's logico-mathematical work Cooper is here issuing a kind of *mea culpa*, but he still takes the mathematician's perspective to be fruitful. "Natural computing", Cooper argues, should be contrasted with the universal Turing machine, which was "unreal" from the beginning.

Cooper adduces a battery of recent claims by scientists—economists, physicists, computer scientists and philosophers—challenging what they regard as a "dogma" about the applicability of Turing's model of computation to physical phenomena. He cites a remark made by Marvin Minsky's at our Turing 100 conference in 2012: "AI is brain dead" because of "the chimera" of embodied intelligence. "Embodiment" is complex: the AI experimental robotic approach fails on fundamental questions, Cooper thinks, in addressing embodied intelligence too directly. The problem is instead really one of higher-order explanations and the "fragmentation" of causality they impose.

Turing himself thought beyond the naïve application of his machine model, pondering complexity, (apparent) emergence, and artificial life in his late works on morphogenesis ((1952); cf. Turing and Ince (2002)). While locally successful, however, Cooper argues that this computational perspective proceeds "pointwise", step by step—like a termite "blindly" building a nest—even though what results, if lying at a height "typed" above the process, is something else, a "phase transition" ultimately deriving from "the underlying information structure". Cooper expresses consternation that Turing "strangely" never returned to his (1938c/2012, 1939) oracle machines, arguing that the multi-leveled approach of "types" could help clarify issues about mental states' causation of, and dependence on, physical events, as well as the whole concept of morphogenesis.

"The confusion" Cooper alleges, is the attempt "to regard causality as being a simpler concept than that of determinism". This will surprise philosophers, who have long regarded causality as multifarious and unclear (Cooper himself quotes Earman here). Distinctions should perhaps be drawn instead between local and global foundational approaches, "closed" vs. "open" systems, degrees of freedom in parametric specifications of empirical systems, and conceptions of objects as entities with fixed predicable properties vs. those that take fundamental elements of nature to be processes that evolve.

It might, in other words, be argued that "determinism", implying fatalism, means that nothing can intervene to change what happens during some system's evolution (so it is "closed"); whereas "causality" has to do, instead, precisely with the possibility of intervention to change a contingent outcome (so it is "open", e.g., to manipulation and hence re-parametrization). From this point of view "determinism" is an article of faith, one that was common in classical physics, viz., that every suitable enlargement of a system may be embedded in a larger one that itself is closed or

"complete"[42]—just as it was an article of faith in the formalistic part of the Hilbert program, pre-Gödel, that foundations could be ultimately be autonomous and finitistic.

Cooper expresses skepticism about the "many worlds" interpretation of quantum mechanics, defending instead the idea of one "hybrid" world of different levels and kinds of computation. This entangles him, toward the end of the paper, in a number of questions about measures of "randomness" in quantum mechanics. Yet ultimately the thickets of causality are deeply intertwined with those of ontology as well as computability. Does the "multiverse" of possible worlds that quantum computers may be thought of as accessing really "exist"? Or do we only have differing processes, characterized locally from one or another standpoint? What *are* "types"?

1.3.3 Human, Machine, and Mind

Part III takes us to the issue of *thinking,* hardly a concept whose contours are sharply delineated for philosophical purposes. Together these essays suggest that Turing may—perhaps ironically—have contributed a great deal to our ability to reach a better understanding of certain aspects of what makes us most distinctive as a species. The question how far our linguistic, narrative, perceptual, social and volitional capacities can be explained by implementation of a computational perspective brings us face to face with the limits of philosophy of mind in its classical form.

In Chap. 11 AI researcher Winston reconsiders the Turing's "Computing Machinery and Intelligence" (1950b) and explains how the AI program has developed since. Stressing that Turing's arguments were designed to overturn philosophical arguments intended to show that computers *cannot* be intelligent, Winston begins with a careful review of their aims. There is a fundamental logical point that is crucial to his exposition: if one wishes to show that something *can* be done—for example, that an algorithm*can* be found (that a machine *can* be intelligent)—one must find and exhibit the positive instance; whereas for negative or limitative results about what *cannot* be done, one must have a precise characterization of what, in general, it *would have been* to do it (witness Turing's account of "computability" in his (1936/7), which delivered a negative result through a characterization of "computable number"). Turing exploits this asymmetry in forwarding his criticisms in (1950b) of a number of different kids of skeptical arguments aiming to establish that machines *cannot* think.

Winston shows how "unthinkability" or "unimaginability" cannot provide an argument to establish a negative result. As he argues, the notion of "intelligence" is what Marvin Minsky called a "suitcase" word, what Wittgenstein would have called a "family resemblance" concept: one into which a number of differing, partly overlapping collections of features of behavior might be placed as "scaffolding" for argument, but without necessarily having definability conditions that can be made

[42] See Stachel (2012) for an elaboration of such a view.

explicit to cover all cases. This would explain our willingness to speak of our phones as "smart" while lacking a general theory of intelligence. Philosophers must take on the burden of explicating the notion of "human intelligence" convincingly and systematically if they are to make their arguments against the possibility of AI stick. Like Cooper, Winston insists on the importance of the fact that "we still lack a computational account of human intelligence"—something we have seen Turing also stating in his early remarks on "judgment" and "intuition" (1936/7, 1938c/2012, 1939/2012). The central importance of Turing's arguments in (1950b) is thus to have accomplished the fundamental negative task of dislodging principled argumentation, rather than directly explaining "intelligence".

Since the famous 1956 conference at Dartmouth inaugurating the field, AI has devoted itself to producing the positive instance of machines exhibiting "intelligent" behavior. Winston points out that the early success stories were partial and "fragile", i.e., not easily transportable to different tasks and situations from the immediate ones targeted. Subsequent successes with "smart" devices have been similar, in that AI has splintered into different sub-fields, "each with its own dogma", each oriented toward immediate applications. Winston seeks a broader perspective, one from which more principled arguments can be derived. The key skills, he hypothesizes, that differentiate human from other species are Turingesque: "the strong story hypothesis" (our ability to narrate, string together and recombine stories) and "the directed-perception hypothesis" (the mechanisms that enable us to direct our perceptual faculties to answer questions, using real and imagined events). Winston argues that an evolutionary shift took place about 60,000 years ago that was crucial: humans began thinking "symbolically". This created the conditions for radical innovation.

These two hypotheses, in Winston's mind, go beyond the Turing context of focusing on the idea of "calculating in a logic" or making ideas intuitive for logical proof: the more general human capacity is that of appreciating and formulating narrative, and/or recipe. The capacity for stringing together narrative is the more powerful, Winston argues, than the capacity for stringing together inferences: crowd-sourcing of narratives on the web is far more efficient than attempting to lodge in a single place a collection of presumed facts from which other facts are to be deduced.

To justify this, Winston describes how he and his students have constructed the "Genesis story understanding system", which uses common sense rules to string together inferences "natural" to human readers of a narrative. Using Shakespeare's *Macbeth*, basic elements of plot are set up and sketched, and the human "craving for understanding" is carried out computationally, as the program builds up an elaboration and explanation for particular events. "Surprisingly", Winston writes, "commonsense rules and concept patterns enable Genesis to answer questions about what happened and why, reflect differing allegiances and cultural points of view, manifest consequences of personality traits, anticipate trouble, determine conceptual similarity, align stories for analogical reasoning, summarize, teach, and persuade". These rules are even culturally relative, as Winston also shows.

This raises a number of interesting questions about Turing's own model, which ultimately depends for its applicability on our ability to use "ingenuity" to develop "surveyable" conceptualizations of brute-force, step-by-step processes. Winston notes two kinds of open questions in his line of AI research. First, there is the daunting issue of sophistication and density in narrative: a simple sketch of plot elements is quite different from a Dickens novel, strewn with metaphors and inner complexities and depths of allusion and meaning, and we are nowhere near automating a machine to put together something this complex. Second, no extant machine is able to extract useful descriptions of human action ("Jane is walking home") from directed perception of an event. We could call this the problem of *depth*, both in symbolic and in perceptual experience. Winston believes that what we require is a better understanding of how our brain actually accomplishes appreciation of depth. This, however, is a long way down the road. Moreover, one suspects that philosophers and humanities scholars are in a position to contribute to characterizing what symbolic density and depth *are*.

In Chap. 12 Rescorla pursues the issue, reconfiguring current argument about whether and in what sense mental processes are Turing-style computations. He takes representational structuring to be crucial to the fundamental structure of human psychology, but contrasts two very different accounts. The first, orthodox one he dubs *the formal-syntactic conception of computation* (FSC), which holds that cognitive computation manipulates signs without regard to their representational or semantic properties. His preferred alternative, the *semantically permeated* (SP) approach, is one on which many core mental computations are composed from inherently representational elements. While Rescorla mentions some of the other paradigms (such as neural networks), his focus is on the classical computational theory of mind, forwarded (and retracted) by Putnam and still defended by Fodor. This is important because FSC is still a focus of influential philosophical critics of computationalism about the mind.[43]

Rescorla argues that the mathematical theory of computation developed by Turing and others does not presuppose any Fodorian "mentalese" about psychological states: Turing's idea was to *replace* the notion of a "state of mind" by a "physical counterpart", an instruction that can be conceived as a "symbolic configuration". The central outstanding psychological problem, argues Rescorla, is to explain *intentionality*, the "aboutness" or "directedness" of mental phenomena of thought and perception first isolated in its modern form by Brentano. This notion has figured as an ideological strike point in philosophy for over a hundred years, shaping debates between continental "phenomenologists" and many of their "analytic" critics, who sought to analyze away the idea of "aboutness" by means of further, external, many-place relations. Among the latter are FSC theorists, who believe that in beginning with syntax—i.e., "purely formal" elements that have no "semantic" or referential or representational interpretation—"intentionality" as a phenomenon need not be taken to be basic.

[43] See Dreyfus (1972, 1992); Searle (1984, 1992).

While Rescorla is prepared to grant that the FSC theory is suitable for personal computers and smart phones, he denies that it can account for computation *in general*. On the view he defends, certain computational models individuate their states through representational properties, and these are what is needed for modeling human language and thought. Tracing his view back to Ockham, Rescorla argues that even though the connection between a particular natural language word and its denotation is "purely arbitrary" ("horse" might have been used, in English, to denote "pony" or "grass")—the word being "semantically neutral"—nevertheless mental states have natural, fixed denotations that cannot be changed "at will". On this picture there is no way to get to something both arbitrary and also explanatory by abstracting all semantical features of a mental state. A concept of ellipse is just that, and its being true of ellipses cannot be abstracted away while leaving it what it essentially is.

The semantic or representational permeation can occur, argues Rescorla, even when mental states that are non-conceptual, as in perceptual states that are attributive (seeing the red tomato as itself red). He commits himself hereby to an "externalist" or "wide content" treatment of representational permeation generally, one which need not always supervene on internal states of neurochemical physiology, but is constituted in part, at least ordinarily, by saturation from the environment. So-called "narrow content", restricted to what does supervene on such internal states, is in his view "semantically indeterminate", though not "semantically neutral" like ordinary words of a natural language. Semantically permeated entities, by contrast, have "intrinsic" meaning, and "lack formal syntactic properties of any theoretical interest"; they are the basic elements out of which representational mental states and processes are composed, analogous to the way in which the secondary colors may be regarded as composed of the primary ones, or nuances of emotion as built upon a few basic physiognomically-detectable facial expressions.

In Chap. 13 Proudfoot pursues the issue of intentionality, considering Turing's attitude toward the traditional problem of freedom of the will and the question of emotions. Metaphysical agency was traditionally a criterion of the divide between the human and the non-human, as in Descartes, who regarded animals as mere "machines" in contradistinction to ourselves. In fact the will solved for Descartes as well the problem of how, given an omniscient and omnipotent God, we are able to make sense of the possibility of human error and the distinction between truth and falsity: though omniscient and omnipotent, God leaves our will "free play" in order that we be able to err and judge. Turing echoes the point in his (1947/1992) when he writes that "if a machine is expected to be infallible, it cannot also be intelligent."

Proudfoot shows that Turing addressed the "controversy" of freedom of the will in every one of his papers on AI. In his (1948c/1992) and (1952) Turing suggested a novel form of *response-dependent* compatibilism in classifying "intelligence" as an "emotional concept". An "emotional concept" is, according to Turing, "determined as much by our own state of mind and training as by the properties of the object under consideration": on this view whether or not an object is classified as "intelligent" is determined "as much by our own state of mind and training as by the properties of the object", and so it is also with our notion of an action "freely willed"

(1948c/1992). Like Hume, skeptical about a rationalistic account of causation in accordance with a general principle of sufficient reason, Turing treated the "problem" of freedom as partially verbal, rather than only or primarily metaphysical and therefore as one whose solution must ultimately reside in "common sense". On this view, part of human nature itself is the tendency to spread readings upon nature by way of psychological associations between the structure of our expectations and classes of observed events. Turing adds to the mix the importance of our working framework of concepts and language. On his view both humans and machines can be the ultimate source of their particular behavior, even if that behavior is also determined.

Proudfoot argues that her reading of Turing informs our understanding of the famed Turing test for "intelligence", justifying its structure. Moreover, it provides a significant corrective to the otherwise widely held view of Turing as a thoroughgoing mechanist and/or behaviorist about the mind. For it casts doubt on readings that have taken Turing's solution to the problem of freedom of the will to have rested wholly on the unpredictable, uncomputable, or indeterministic outcomes of equations in physics. Turing did not think that merely equipping a digital computer with a "random" element would suffice for it to "act freely". The robust notion of "freedom" at stake must be able to make sense of the idea that an action is free if and only if the agent *could have acted differently* or is the *ultimate origin* of the action. Proudfoot's reading brings into view the possibility of regarding Turing's conception of mentality as far more open to plasticities in the structuring and restructuring of evolving human patterns of agency and action than the reductive mechanist picture allows.

In Chap. 14 Sterrett brings to the forefront Turing's emphasis on the nature of differing kinds of *searches*, focusing on the *integration* of machine and human intelligence. In his post-war work Turing moved out into the frontiers, not only of computing, but of science and human life itself. He not only envisioned a simplification of notations to generate further mathematical theorems, the mastery of mathematics for scientific and military applications, and the use of computation to resolve outstanding conjectures, or parlor game computer contests. He also looked forward to the embedding of routines of the stored program computer in every facet of our experience, among other things to augment and enhance human life itself, partly by exploring the idea of a brain as a computational device instantiating his universal machine model, and partly by marshaling the force and structure of human cultural evolution.

Sterrett sees the usual emphasis on Turing's (1950b) *Mind* paper as unfortunate. Turing's thinking was, she argues, far more forward-looking and sophisticated than readers of that essay might think, and she considers at length two recent success stories that she believes vindicate Turing's ideas and utilize important elements of his other ideas. First Sterrett considers Carnegie Mellon's NELL (Never Ending Language Learning), a part of the so-called "semantic web" which continuously "learns" by "reading" massive amounts of material derived from millions of web pages, cutting out combinatorial pieces of information to attempt to get to the heart of a taxonomy through analysis of basic ones. Next she explains IBM's "Watson",

whose lightning-quick question-answering capacities made it capable of winning the game show *Jeopardy!* in 2011. Her point is that both these ventures rely on human guidance and feedback at various points in the machine's learning process, although the manner in which Watson targeted its searches, basing them on keywords gleaned from Wikipedia, proved far superior for its task, and largely because Wikipedia itself reflects *prior* human effort.

As Sterrett emphasizes, Turing was quite explicit that that "an isolated [human] does not develop any intellectual power" (1948c/1992). He relied on the presence of speech and trial and error learning in his model, which utilized a sophisticated battery of methods of search taking the surroundings as important, rather than subjecting an individual machine to programming of trial and error games in isolation. In fact, human interaction with the machine, and human to human interaction as a whole, are crucial touchstones for Turing's vision of machine learning, and this throughout the whole developmental process. Turing pointed out, for example, that the human ability to pick up on an unusual rule (e.g. an irregular verb) need not be fully built-in from the start, but must be taken as a residual basis on which the entire evolutionary process counts.

To this end Turing emphasizes three kinds of search: *intellectual*, *genetic* (or evolutionary), and *cultural*. "Intellectual" searches aim to resolve problems into elements that are then subject to mathematical structuring with parameters; this could involve human and machine cooperation, as in the automated theorem proving based on a logical system such as *Principia Mathematica*, which, while formalized, nevertheless demands a high degree of intellectual power to carry out and devise proofs. Sterrett insists that even this kind of search involves the interaction of individual humans in society: *Principia* stores up a great deal of knowledge it has inherited from the past. "Genetical" or "evolutionary" searches were inspired, she argues, by Turing's appreciation of the new biological theories of genetic modification injected into evolutionary theory by 1948, the time when Turing wrote (1948c/1992). These searches make plausible the relevance of intellectual searches through the combinatorial possibilities to predict "selective advantage", but those involving "new techniques" require, Turing argues, frequent "human to human communication", a wider connection with training and culture.

In Chap. 15 Parikh returns to logic to consider the idea of a "Church-Turing" thesis for social algorithms. Aspects of many everyday situations can be modeled by a particular series of formalisms that Parikh has developed, each of which serve to "constructivize" or "Turingize" the operational aspects of many different social and cognitive events, using a flexible, multidimensional model of preferences among possible worlds, agent relations, utilities, and character types. If cognition is conceived as a process happening among many agents, we must be partly concerned with the design, construction and understanding of possible systems for society. This is a new way of approaching the question of precisely *how* processes of semantic permeation and "depth" should be conceptualized at the fundamental level. Parikh suggests that the constructive elements of logical theory are there to be used

in elaborating a better understanding of cognition and society from the perspective of what he calls "social software".[44]

Invoking philosopher-economist Sen's "capabilities" approach, Parikh points out that rational choice theorists for too long abstracted away from the actual opportunity and belief spaces of agents, forgetting normative and situated issues involving agential perspectives. Society is not simply one fixed contract between two equal parties, but a series of structured situations in which some rules are announced, others hidden, and others brought onto the table and then set aside. As Parikh puts it, "the dance between knowledge and ignorance drives much social action". Presenting the logical structure of several different games, Parikh demonstrates that the *way* in which a particular piece of information is shared and *with whom* and *at what point* have a lot to do with the *content* and *usefulness* of that information, semantically speaking, in a given situation—and vice versa. So far as the logic is concerned, in real life it is not a matter of a total-information space, but of a complex "dance" of positioning and lack of positioning, announcing and holding mum—all the complex of speech acts which have been emphasized since J.L. Austin, Wittgenstein and Grice as part of what allows us to individuate particular truth conditions and communicate our desires through *recognized* procedures.

Knowledge means and is influenced by power, and is held by groups as well as individuals. Tigers appear to have been able to develop a theory of "false belief" over time, working as a group to overcome human efforts to control and deceive them. Chimps take into account the epistemic situation of other chimps in ferreting out food. Parikh shows how many everyday actions, from royal processions to waiting on tables or getting married, involve complex but systematic structures of knowledge: the retrieval, manipulation, public expression (sometimes silence about) knowledge and ignorance as they are understood by participants. Each of these structure the semantic richness of those elements chosen as fundamental to the situation.

Narratives are relevant here as well: Parikh reconstructs the plot of Shakespeare's *Much Ado About Nothing* using a logic of knowledge to show how commonly held knowledge and its generators may bring about or hinder our social and emotional "dances" and couplings with one another. The potential infinity of common knowledge—that *i* knows that *j* knows that *i* knows and so on—signals the need for serious conceptual and logical work at structuring such situations.[45] Developing an axiomatization of the logic of knowledge, Parikh shows how such a theory may be rigorously articulated mathematically, and suggests that it may even be relevant to theorizing non-human investigations of social cognition and "theory of mind".

[44] See Parikh (2001, 2002), Pacuit and Parikh (2006).

[45] For an overview of some of the issues, see Hendricks and Symons (2014).

1.3.4 Concluding Précis

In sum: the three parts of this volume show the philosophical legacy of Turing as it emerged from its roots in his earliest research. From the foundations of logic he blueprinted the stored program computer concept, holding on to "common sense" and simplicity as touchstones. Inspired by the Universal Machine, he devised new forms of encryption and production of voice and music, opening up the prospects of quantum computing and post-Turing analyses of types and computations. Finally, turning us back to ourselves, he exposed aspects of our concepts of intelligence that we had not appreciated before, enabling their computational implementation and analysis.

Human mentality, cultural and societal searches themselves will continue to evolve within a new order of human and machine integration thanks to Turing's work.

1.4 Bibliographical Remarks

Modes of reference to Turing's papers differ from essay to essay due to several reprintings of Turing's work, often with useful commentary. Here we discuss the most prominent places where Turing's work is presented and discussed.

A user-friendly complete bibliography is currently kept by Andrew Hodges, who wrote a magnificent biography of Turing (Hodges 1983/2012). Hodges also maintains a useful website categorizing Turing's work.[46] A recurrently updated BibTex bibliography with a guide to websites, a bibliography of Turing's works, and references to papers about Turing's work, is maintained by Professor Nelson H.F. Beebe.[47] B. Jack Copeland and Diane Proudfoot, directors of the Turing Archive for the History of Computing at the University of Canterbury, New Zealand, have also developed a Britannica Internet Guide award-winning site.[48]

A digital archive of Turing's papers is maintained by King's College Cambridge.[49]

Turing's papers (some unpublished) were collected in four volumes with commentary.[50] Petzold (2008) gives a line-by-line commentary on Turing's famous (1936/7)

Two important collections of Turing's reprinted papers with extensive commentary by experts have appeared in recent years: (Copeland (2004) and Cooper and van Leeuven (2013). As to the continued historical, philosophical, and scientific analysis of the impact of Turing's work, there are several volumes that are especially interesting. Herken (1998) surveys the impact and legacy of Turing's Universal Machine. Teuscher (2004) discusses Turing's life and legacy. Church's Thesis is the

[46] http://www.turing.org.uk/sources/biblio.html, http://www.turing.org.uk/index.html

[47] http://ftp.math.utah.edu/pub//bibnet/authors/t/turing-alan-mathison.html

[48] www.alanturing.net

[49] http://www.turingarchive.org/about/

[50] Turing and Britton (1992), Turing and Ince (1992), Turing and Saunders (1992), Turing and Gandy and Yates (2001)).

focus of Olszewsky, Woleński and Janusz (2006). Essays analyzing the notion of computability may be found in Copeland, Posy and Shagrir (2013), while further analysis of Turing's historical situation and the scientific legacy of his ideas are treated in Sommaruga and Strahm (2015). Deepening historical analyses of Turing's specific contributions to the history of computing may be found in Hodges (2003), Copeland (2006), Copeland (2011a, b, 2012).

Among new institutes of research that have recently sprung up are the Turing Gateway to Mathematics, aiding users of mathematics across the sciences, housed at the Isaac Newton Institute for Mathematical Sciences, Cambridge University[51], the Alan Turing Institute for Data Science, housed at Britain's National Library in London[52], and the Turing Centre, Zürich, a collaborative research and teaching center for "Turing Studies", i.e., foundations of computing, history of computing, AI, A-Life, neurocomputing, quantum computing and hypercomputating, philosophy of computing, history and philosophy of mathematics, cryptography, mathematical logic and mathematical biology. "Computability in Europe" has spawned annual conferences, research series, and journals.

The site maintained by S. Barry Cooper in honor of Turing's centennial, "The Alan Turing Year", holds a trove of information about intellectual, cultural and scientific events that continue to celebrate his life.[53] The flow of papers influenced by Turing is continually growing, and hence the assessment of his impact, on which one may see the *Nature* feature "Turing at 100: A Legacy that Spans Science (nature.com/turing).[54] There is also the "Turing law", passed by the British government in October 2016, and extended in February 2017, opening the way up for a mass pardoning of those roughly 65,000 individuals convicted for homosexuality under laws since repealed.

As to primary sources, under Britain's Official Secrets Act, many papers relating to Turing's work at Bletchley Park (1939–1945) were not released for 50 years after the war, and more papers are expected to be released in the future.

Acknowledgments The author gratefully acknowledges comments on a late draft by Juliette Kennedy and Alisa Bokulich that crucially improved this Introduction, as well as John Stachel's help in sharing helpful feedback on aspects of the papers entangled with physics and causality.

References

Becher, V., S. Figueria, and R. Picchi. 2007. Turing's unpublished algorithm for normal numbers. *Theoretical Computer Science* 377: 126–138.

Blum, L. 2013. Alan Turing and the Other Theory of Computation, Commentary on Turing (1948a) in Cooper and van Leeuven (eds.) 377–384.

Brooks, R. 2001. The Relationship Between Matter and Life. *Nature* 409: 409–411.

[51] http://www.turing-gateway.cam.ac.uk/about.shtml

[52] https://turing.ac.uk/

[53] http://www.mathcomp.leeds.ac.uk/turing2012/

[54] Cf. also Brooks (2001).

Church, A. 1936. An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics* 58: 345–363.

———. 1937. Review of Turing, "On Computable Numbers, with an Application to the *Entscheidungsproblem*", *The Journal of Symbolic Logic* 2: 42–43. Reprinted with Commentary by A. Hodges in Cooper and van Leeuven (eds.) (2013), 117–120.

Cooper, B.S., and van Leeuven (eds.). 2013. *Alan Turing: His Work and Impact*, Amsterdam/ Waltham: Elsevier.

Copeland, B.J. 2009. Spring edition of *The Stanford Encyclopedia of Philosophy*, "Church-Turing": http://plato.stanford.edu/archives/spr2009/entries/church-turing/.

———. 2011a. The Manchester Computer: A Revised History, Part I: The Memory. *IEEE Annals of the History of Computing* 33 (1): 4–21.

———. 2011b. The Manchester Computer: A Revised History, Part 2: The Baby Computer. *IEEE Annals of the History of Computing* 33 (1): 22–37.

———. 2012. *Turing: Pioneer of the Information Age*. Oxford: Oxford University Press.

———. (ed.). 2004. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life*. Oxford, Clarendon Press.

———. (ed.). 2006. *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*. Oxford: Oxford University Press.

Copeland, B.J., C.J. Posy, and O. Shagrir (eds.). 2013. *Computability: Turing, Gödel, Church, and Beyond,* Cambridge, MA: MIT Press.

Davis, M. 1982. Why Gödel Didn't Have Church's Thesis. *Information and Control* 54: 3–24.

———. 1987. Mathematical Logic and the Origin of Modern Computers. In *Studies in the History of Mathematics*, 137–165. Washington, DC: Mathematical Association of America. Reprinted in Herken (ed.) (1988), 149–174.

———. 2000/2011. *The Universal Computer: The Road from Leibniz to Turing*. New York: Norton, AK Peters, CRC, Taylor & Francis.

Davis, M., ed. 1965/2004. *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. New York: Raven Press/Dover Publications.

Davis, M., and W. Sieg. 2015. Conceptual Confluence in 1936: Post and Turing, in G. Sommaruga and T. Strahm (eds.), 3–28.

Detlefsen, M. 1986. *Hilbert's Program: An Essay on Mathematical Instrumentalism*, Springer Library. Springer: Dordrecht.

Dreyfus, H.L. 1972. *What Computers Can't Do: A Critique of Artificial Reason*. New York: Harper & Row.

———. 1992. *What Computers Still Can't Do: A Critique of Artificial reason*. Cambridge, MA: MIT Press.

Dyson, G. 2012. *Turing's Cathedral: The Origins of the Digital Universe*. New York: Pantheon Books.

Feferman, S. 1962. Transfinite Recursive Progressions of Axiomatic Theories. *The Journal of Symbolic Logic* 27 (3): 259–316.

———. 1988. Turing in the Land of O(z), in R. Herken (ed.) (1988), 113–147.

Franks, C. 2009. *The Autonomy of Mathematical Knowledge: Hilbert's Program Revisited*. Cambridge: Cambridge University Press.

Franzén, T. 2005. *Gödel's Theorem: An Incomplete Guide to Its Use and Abuse*. A.K. Peters: Wellesley.

Gandy, R.O. 1988. The Confluence of Ideas in 1936, in Herken, R. (ed.), (1988), 55–112.

Gettier, E.L. 1963. Is Justified True Belief Knowledge? *Analysis* 23 (6): 121–123.

Gödel, K. 1930. The Completeness of the Axioms of the Functional Calculus of Logic. In *Kurt Gödel Collected Works Volume I: Publications 1929-1936*, eds. S. Feferman, et.al., 103–123. New York/Oxford:Oxford University Press, 1986. Original publication "Die Vollstandigkeit der Axiome des logischen Funktionenkalküls", *Monatshefte fur Mathematik und Physik* 37: 349–360.

———. 1931. On Formally Undecidable Propositions of *Principia Mathematica* and Related Systems I. In *Kurt Gödel Collected Works: Vol. I, Publications 1929–1936*, ed. S. Feferman et.al., 145–195. New York: Oxford University Press, 1986.

———. 1934. On Undecidable Propositions of Formal Mathematical Systems. In *Kurt Gödel Collected Works, Vol. I: Publications 1926–1936*, ed. S. Feferman, 346–372. New York/Oxford: Oxford University Press.

———. 1946. Remarks Before the Princeton Bicentennial Conference on Problems in Mathematics. In *Kurt Gödel Collected Works, Vol. II: Publications 1938–1974*, eds. S. Feferman et.al., 150–153. New York: Oxford University Press, 1990.

———. 1956. Letter to von Neumann, 20 March 1956, in *Kurt Gödel Collected Works. V: Correspondence H-Z*, eds. S. Feferman et al., 372–375. Oxford: Oxford University Press, 2003.

———. 1964. Postscriptum to the 1934 Princeton Lectures. In *Kurt Gödel Collected Works Vol. I: Publications 1929–1936*, eds. S. Feferman, et al., 369–370. New York: Oxford University Press, 1986.

Good, I.J. 1979. A.M. Turing's Statistical Work in World War II, Studies in the History of Probability and Statistics. XXXVII. *Biometrika* 66, 2, 393–396. Reprinted in Turing and J.L. Britton (eds.) 1992, 207–210.

———. 2000a. Turing's Anticipation of Empirical Bayes in Connection with the Cryptanalysis of the Naval Enigma. *Journal of Statistical Computation and Simulation* 66: 101–111.

———. 2000b. Introductory Remarks for Good (1979), in Turing and Britton J. (eds.) (1992), 211–223.

Grattan-Guinness, I. 2013 The Mentor of Alan Turing: Max Newman (1897–1984) as a Logician. *The Mathematical Intelligencer* 35: 54–63.

Hilbert, D. 1900. Mathematische Probleme. Vortrag, gehalten auf dem internationalen Mathematiker-Kongress zu Paris. *Nachrichten der Königlichen Gesellschaft der Wissenschaften zu Göttingen* 253–297. Reprinted, with additions, in *Archiv der Mathematik und Physik* series 3, 1 (1901), 44–63, 213–237. Translated by Mary Winston Newson in *Bulletin of the American Mathematical Society* Series 2, 8 (1902), 437–479; Selections Reprinted as 1900b in *From Kant to Hilbert*, Vol. 2, ed. W. Ewald, 1096–1105. Oxford/New York: Oxford University Press.

Herken, R. (ed.). 1988. *The Universal Turing Machine: A Half-Century Survey.* Oxford/New York: Oxford University Press.

Hendricks, V., and J. Symons. 2014. Epistemic Logic. *Stanford Encyclopedia of Philosophy*, spring 2014, at http://plato.stanford.edu/archives/spr2014/entries/logic-epistemic/.

Hinsley, E.H., and A. Stripp (eds.). 1993. *Codebreakers: The Inside Story of Bletchley Park*. Oxford: Oxford University Press.

Hodges, A. 1983/2012. *Alan Turing the Enigma of Intelligence*. New York/Princeton: Touchstone/Princeton University Press.

———. 2003. The Military Use of Alan Turing. In *Mathematics and War*, B. Boß-Bavnbek and J. Høyrup (eds.), 312–325. Boston: Birkhäuser.

———. 2013. Commentary on Church (1937) in Cooper and van Leeuven (eds.), (2013), 117–118.

Jones, C.B. 2013. Turing's (1949) 'Checking a Large Routine', in Cooper and van Leeuven (eds.), (2013), 455–460.

Kahn, D. 1991. *Seizing the Enigma: The Race to Break the German U-Boat Codes, 1939–1943*. Boston: Houghton Mifflin Company.

———. 1996. *The Codebreakers: The Story of Secret Writing* (Revised ed.). New York: Scribner.

Kleene, S.C. 1981a. Origins of Recursive Function Theory. *Annals of the History of Computing* 3 (1): 52–67.

———. 1981b. The Theory of Recursive Functions, Approaching its Centennial. *American Mathematical Society, Bulletin, New Series* 5 (1): 43–61.

Moor, J.H. (ed.). 2003. *The Turing Test.* Dordrecht: Kluwer.

Newman, M.H.A. 1955. Alan Mathison Turing. 1912–1954. *Biographical Memoirs of Fellows of the Royal Society* 1: 253–263.

Olszewski, A., J. Wolenski, and R. Janusz (eds.). 2008. *Church's Thesis After 70 Years*. Ontos Mathematical Logic. Frankfurt: Ontos.

Oppy, G., and D. Dowe. 2011. The Turing Test. *Stanford Encyclopedia of Philosophy* spring 2011, at http://plato.stanford.edu/archives/spr2011/entries/turing-test/.

Pacuit, E., and R. Parikh. 2006. Social Interaction, Knowledge, and Social Software. In *Interactive Computation: The New Paradigm*, D. Goldin, S. Smolka, and P. Wegner (eds.). New York: Springer.

Parikh, R. 2001. Language as Social Software. In *Future Pasts: The Analytic Tradition in Twentieth Century Philosophy*, J. Floyd and S. Shieh (eds.), 339–350. New York: Oxford University Press.

Petzold, C. 2008. *The Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine*. Indianapolis: Wiley Publishing, Inc..

Plato. 1997. *Complete Works*, ed. John M. Cooper. Indianapolis: Hackett Publishing.

Post, E.L. 1936. Finite Combinatory Processes, Formulation 1. *The Journal of Symbolic Logic* 1: 103–105.

Putnam, H. 1988. *Representation and Reality*. Cambridge: MIT Press.

Russell, B. 1912/1990. *The Problems of Philosophy*. Indianapolis, Hackett Publishing Co. Originally published by The Home University Library of Modern Knowledge.

Searle, J.R. 1984. *Minds, Brains and Science: The 1984 Reith Lectures*. Cambridge, MA: Harvard University Press.

———. 1992. *The Rediscovery of Mind*. Cambridge, MA: MIT Press.

Shieber, S. (ed.). 2004. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence.* Cambridge, MA: MIT Press.

Sieg, W. 2009. On Computability. In *Philosophy of Mathematics*, Handbook of the Philosophy of Science Series, ed. A.D. Irvine, 535–630. Amsterdam/Boston/Oxford: Elsevier.

———. 2013. *Hilbert's Programs and Beyond*. Oxford/New York: Oxford University Press.

Stachel, J. 2012 Where is Knowledge?. In *Frontiers of Fundamental Physics/the Eleventh International Symposium, AIP Conference Proceedings*, ed. Joseph Kouneiher et al., vol. 1446, 312–334. Melville: American Institute of Physics.

Sommaruga, G., and T. Strahm (eds.). 2015. *Turing's Revolution: The Impact of His Ideas About Computability.* Basel: Birkhäuser/Springer.

Turing, A.M. 1935a. On the Gaussian Error Function. Unpublished Fellowship Dissertation, King's College Library, Cambridge, available in the Turing Digital Archive as AMT/C/28 at http://www.turingarchive.org/browse.php/C/28.

———. 1935b. Equivalence of Left and Right Almost Periodicity. *Journal of the London Mathematical Society* 10: 284–285. Reprinted in Turing and Britton (eds.) (1992), 1–2.

———. 1936/1937. On Computable Numbers, with an Application to the Decision Problem. In *Proceedings of the London Mathematical Society* 2(42): 230–265. Correction: "On Computable Numbers, with an Application to the Decision Problem: A Correction" (1937), Proceedings of the London Mathematical Society, 2 (42) (1937), 230–265. Both reprinted with Commentary in Davis (ed.) (1965/2004), 115–153; Turing, and Gandy, R.O. and Yates, C.E.M. (eds.), (2001), 18–53; Copeland, J. (ed.) (2004), 58–90; Petzold, C. (2008); Cooper and van Leeuven eds. (2013), 16–41.

———. 1937a. Computability and λ-definability. *Journal of Symbolic Logic* 2: 153–163. Reprinted in Turing and Gandy and Yates (eds.) (2001), 59–69.

———. 1937b. The p-function in λ-*K* Conversion. *The Journal of Symbolic Logic* 2 (1937): 164.

———. 1938a. Finite Approximations to Lie Groups. *Annals of Mathematics, Princeton* 39: 105–111. Reprinted in Turing and Britton (eds.) (1992), 3–10.

———. 1938b. The Extensions of a Group. *Compositio Mathematica* 5: 357–367. Reprinted in Turing and Britton (eds.) (1992), 11–22.

———. 1938c/2012. Systems of Logic Based on Ordinals. Dissertation, Princeton University 1938. Published in *Alan Turing's Systems of Logic: The Princeton Thesis* (ed.) A. Appel (Oxford and Princeton, Princeton University Press).

———.1939. Systems of Logic based on Ordinals. *Proceedings of the London Mathematical Society* 2(45): 161–228. Reprinted in Turing and Gandy and Yates (eds.) (2001), 81–148.

———. 1941. Some Theorems About Church's System. Unpublished manuscripts, in Turing and Gandy and Yates (eds.) (2001), 201–206.

———. 1942. The Use of Dots as Brackets in Church's System. *The Journal of Symbolic Logic* 7: 146–115.

———. c.1940. "Treatise on the Enigma", Known as "Prof's Book". Xerox TS, Part of the records of the National Security Agency transferred to US National Archives and Records Administration in Apr. 1995. Paper, 61 sh. in envelope. Available online at the Turing Digital Archive (http://www.turingarchive.org/browse.php/C/30) as AMT/C/30. Excerpted with commentary in Turing and Gandy and Yates (eds.) (2001) 223–258 and in Cooper and van Leeuven (eds.) (2013), 413–425. Ch. 6 reprinted with related correspondence and commentary in Copeland 2004, 311–352.

———. 1943. A Method for the Calculation of the Zeta-Function. *Proceedings of the London Mathematical Society* 2(48): 180–197. Reprinted in Turing and Britton (eds.) (1992), 23–40.

———. 1943/1944. Practical Forms of Type Theory II. Unpublished paper in Turing and Gandy and Yates (eds.) (2001), 207–211.

———. 1944/1945. The Reform of Mathematical Notation and Phraseology. Unpublished paper in Turing and Gandy and Yates (eds.) (2001), 211–222. Reprinted in Cooper and van Leeuwen (eds.) (2013), 245–249.

———. 1945/1992. Proposals for development in the Mathematics Division for an Automatic Computing Engine (ACE), Report to the Executive Committee of the National Physics Laboratory, in Turing and D.C. Ince (ed.) (1992), 1–86. Originally published in B.E. Carpenter and R.N. Doran (eds.), *A.M. Turing's ACE Report of 1946 and Other Papers*, Cambridge, MA 1986, 20–105.

———. 1947/1992. Lecture on the Automatic Computing Engine, given to the London Mathematical Society 20 February 1947, in Turing and Ince (eds.) (1992), 87–106. Originally published in B.E. Carpenter and R.N. Doran (eds.), *A.M. Turing's ACE Report of 1946 and Other Papers*, Cambridge, MA 1986, 106–124.

———. 1948a/1992. Rounding-off Errors in Matrix Processes. *Quarterly Journal of Mechanics and Applied Mathematics* 1: 287–308. Reprinted in Turing and Britton (eds.) (1992), 41–62.

———. 1948b. Practical Forms of Type Theory. *The Journal of Symbolic Logic* 13: 80–94. Reprinted in Turing and Gandy and Yates (eds.) (2001), 179–202.

———. 1948c/1992. "Intelligent machinery", in Turing and Ince (ed.) (1992), 107–128. Report, National Physics Laboratory, originally published in B. Meltzer and D. Michie (eds.), *Machine intelligence* 5. Edinburgh: Edinburgh University Press, 1969, 3–23.

———. 1949. Checking a Large Routine. EDSAC Inaugural Conference 24 June 1949, Report of a conferernce on High Speed Automatic Calculating Machines: 67–69. Reprinted in Turing and Ince (eds.) (1992), 87–106.

———. 1950a. The Word Problem in Semi-groups with Cancellation. *Annals of Mathematics* 52 (2): 491–505. Reprinted in Turing and Britton (eds.) (1992), 63–78.

———. 1950b. Computing Machinery and Intelligence. *Mind* 59(October): 433–460. Reprinted in Turing and Ince (eds.) (1992), 133–160.

———. 1951a. Intelligent Machinery, A Heretical Theory, BBC, The '51 Society Radio Broadcast. The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life. B. S. Cooper. Oxford: Oxford University Press, 465–475.

———. 1951b. Can Digital Computers Think? BBC Radio Broadcast, 15 May and 3 July 1951. Published in Copeland (2004), 476–486.

———. 1952. The Chemical Basis of Morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* 237 (651): 37–52. Reprinted in Turing and Saunders (eds.) (1992), 1–36.

———. 1953a. Some Calculations of the Riemann Zeta-Function. *Proceedings of the London Mathematical Society* 3 (3): 99–117. Reprinted in Turing and Britton (eds.) (1992), 79–98.

———. 1953b. Digital Computers Applied to Games: Chess. In *Faster Than Thought*, ed. B.V. Bowden, 286–310. London: Pitman.

———. 1954. Solvable and Unsolvable Problems. *Science News* 31: 7–23. Reprinted in Turing and Britton (eds.) (1992), 99–116 and in Turing and Ince (eds.) (1992), 187–204.

Turing, A.M. ?. *On Permutation Groups*, in Turing and Britton (eds.) (1992), 125–146.

———. 195?. Turing, A. M. *Programmers' Handbook for Manchester Electronic Computer Mark II*, Computing Machine Laboratory, University of Manchester, no date, circa 1950. A digital facsimile is in *The Turing Archive for the History of Computing* at www.AlanTuring.net/programmers_handbook. Third edition prepared by Tony Brooker in 1953.

Turing, A.M., and M.H.A. Newman. 1942. A Formal Theorem in Church's Theory of Types. *The Journal of Symbolic Logic* 7: 28–33.

Turing, A.M., R. Braithwaite, G. Jefferson, and M.H.A. Newman. 1952. Can Automatic Calculating Machines be Said to Think? BBC Radio Broadcast, 10 January 1952. Published in Copeland (2004), 486–506.

Turing, A.M., and J.L. Britton (eds.). 1992. *Collected Works of A.M. Turing: Pure Mathematics* Amsterdam/New York: Elsevier.

Turing, A.M., and D. Ince (eds.). 1992. *Collected Works of A.M. Turing: Mechanical Intelligence.* Amsterdam/New York: Elsevier.

Turing, A.M., and P.T. Saunders (eds.). 1992. *Collected Works of A.M. Turing: Morphogenesis.* Amsterdam/New York: Elsevier.

Turing, A.M., R.O. Gandy, and C.E.M. Yates (eds.). 2001. *Collected Works of A.M. Turing: Mathematical Logic*, Amsterdam/New York: Elsevier.

Turing, S. 1959/2012. *Alan M. Turing, Centenary Edition.* Cambridge University Press.

von Neumann, J. 1932/1968. *Mathematische Grundlagen der Quantum Mechanik*. Berlin: Springer.

Wilkinson, J. 1971. Some Comments from a Numerical Analyst. *Journal of the Association for Computing Machinery* 18 (2): 137–147.

Wittgenstein, L. 1979. In *Wittgenstein's Lectures on the Foundations of Mathematics: Cambridge, 1939*, ed. R.G. Bosanquet, N. Malcolm, R. Rhees, Y. Smythies, and C. Diamond. Chicago: University of Chicago Press.

Zabell, S.L. 1995. Alan Turing and the Central Limit Theorem. *The American Mathematical Monthly* 102 (6): 483–494.

———. 2013 Alan Turing and the Central Limit Theorem, in Cooper and van Leeuwen eds. (2013), 257–263.

Part I
Logic and Mathematics to Philosophy

Chapter 2
Turing, the Mathematician

Daniele Mundici and Wilfried Sieg

Abstract Turing was awarded the Order of the British Empire in June of 1946. Most people thought that the award was a well deserved mark of recognition honoring the mathematician who had given a successful definition of *mechanical procedure*, had introduced the *universal machine* capable of simulating all mechanical procedures, and had settled in the negative Hilbert's *Entscheidungsproblem*, the problem whether mathematics is fully mechanizable. Few knew that Turing had received the OBE mainly because of his outstanding services as a cryptographer at Bletchley Park during the Second World War.

This essay is about Turing's *mathematical* achievements. We will, as informally as possible, analyze his 1936 paper "On computable numbers with an application to the *Entscheidungsproblem*" and highlight the main ideas contained in this classical paper. Naturally, we will also discuss the broader context of Turing's definitions and theorems, their fundamental interaction with Gödel's completeness and incompleteness theorems, and their basic role in the proof of the Cook-Levin NP-completeness theorem and in the formulation of the P/NP problem.

2.1 Solving the *Entscheidungsproblem*

In the spring of 1935, Turing attended a course on the foundations of mathematics that was given in Cambridge by the algebraic topologist Maxwell Newman. He learned that the deduction rules isolated from mathematical practice and meticulously formalized by Frege (Whitehead, Russell, Hilbert, Bernays, and others) are complete: from any set of premises no new logical consequences can be drawn beyond those already obtainable by applying the classical rules. This is Gödel's completeness theorem for first-order logic. (In (Mundici 2012) one can find an

D. Mundici (✉)
Department of Mathematics and Computer Science "Ulisse Dini", University of Florence, Viale Morgagni 67/A, 50134 Florence, Italy
e-mail: mundici@math.unifi.it

W. Sieg
Department of Philosophy, Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: sieg@cmu.edu

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_2

elementary proof.) He learned also that formal frameworks for mathematics, such as Whitehead and Russell's system of *Principia Mathematica* (PM) or Zermelo-Fraenkel set theory, if consistent, contain a true sentence that cannot be proved in that system. This is Gödel's first incompleteness theorem for the specific formal systems mentioned and holds even for Peano arithmetic. Finally, he learned that, despite these two results, Hilbert's *Entscheidungsproblem* was still open. Several people (including Hardy, Weyl, and von Neumann) had expressed the view that mathematics is more art than computation, whence mathematical problems cannot be perfunctorily settled in a finite number of (mechanical) computation steps—but how can one define "computation step"?

Attracted by this conceptual problem, Turing wrote a masterpiece in which he solved the problem by introducing a theoretical mechanical device that is now called a *Turing machine*. Relative to these machines, he defined the notion of "computation step" and that of a "mechanical procedure"; this was the necessary first step in finding a negative solution of the *Entscheidungsproblem*. Notice, however, that unlike solvability of equations by radicals or constructibility of regular *n*-gons by ruler and compass, such a notion was to encompass all past, present, and future notions of algorithmic operations carried out by a (*human*) *computor*. (As to the choice of this terminology, see Sect. 2.3 on *Mechanical Procedures*.) When compared to the mechanical devices developed by, among others, Pascal, Leibniz, and Babbage, Turing machines have two distinguishing features:

(i) *any* algorithm a computor can carry out can be computed by some Turing machine,
and
(ii) a *single*, so-called *universal* Turing machine can simulate *any* Turing machine on any input.

Turing explicitly constructed such a universal machine, and the underlying idea turned out to be crucial for the architecture of modern computers. In the very same year as Turing, Church also solved the *Entscheidungsproblem* in the negative using a quite different mathematical concept to characterize computation step and mechanical procedure. (For Church's considerations, see (Sieg 1997).)

Having full confidence in the generality of the rigorous notion of mechanical procedures in terms of his machines, i.e., what is now called *Turing's Thesis*, Turing settled the *Entscheidungsproblem* in three steps:

1. He first introduced a problem that cannot be decided by any Turing machine. This is easy, because Turing machines are nothing but suitable finite strings of symbols of some fixed finite alphabet, so they can be listed lexicographically. Let T_1, T_2,... be the lexicographic list of all Turing machines. Then the problem whether machine T_n will eventually stop over input *n* (printing a 0) is Turing-undecidable. No machine T can output "yes" or "no" according as T_n does or does not stop over input *n*. For otherwise, we can slightly modify T into a machine T* that stops on input *n* if and only if T_n does not stop over input *n*. Now, T* appears somewhere in the list of all Turing machines, say $T^*=T_{n^*}$. What

will T* do over input *n**? A moment's reflection shows that T* over input *n** stops if and only if T* does not stop over input *n**—a contradiction. Consequently, a machine T that solves the *halting* or *printing problem* does not exist.

2. He then meticulously translated the phrase "T_n over input *n* eventually halts" into a formula F_n of first-order logic. He gave details of F_n in §11 of his paper (and additional ones in a subsequent note of errata for the same journal in which the original paper had been published). Given the broad compass of his machines and the effectiveness of the lexicographic ordering of these machines, Turing had no doubt that some Turing machine over input *n* would be able to output F_n. In short, Turing constructed a *Turing-reduction*, i.e., a Turing-computable map from *n* to F_n such that T_n over input *n* eventually stops if and only if F_n is a logical truth, i.e., by Gödel's completeness theorem, if and only if F_n is provable. Here are Turing's own words (Turing 1936):

> Corresponding to each computing machine M we construct a formula Un(M) and we show that, if there is a general method for determining whether Un(M) is provable, then there is a general method for determining whether M ever prints 0.

3. Finally, he concluded that the Entscheidungsproblem is unsolvable. Indeed, if the logical truths in first-order logic were decidable by a "general (mechanical) process", then, Turing maintained, the logical truths would also be decidable by some Turing machine D. This machine D, however, would allow us to resuscitate the above machine T in a new form: T would just wait for the response of D to the question "Is F_n a logical truth?" and reply "Yes", if D determines that F_n is a logical truth, and "No", if D determines that it is not. This T answers the question whether T_n halts on input n and prints 0. Such a machine is however impossible and, consequently, D cannot exist and Hilbert's Entscheidungsproblem has a negative solution. In Turing's own words:

> We are now in a position to show that the *Entscheidungsproblem* cannot be solved. Let us suppose the contrary. Then there is a general (mechanical) process for determining whether Un(M) is provable. By Lemmas l and 2 this implies that there is a process for determining whether M ever prints 0, and this is impossible, by §8. Hence the *Entscheidungsproblem* cannot be solved.

The early development towards Turing machines and related concepts of computability was presented already in (Sieg 1994) and in our (1995); in the next two sections we re-emphasize the crucial conceptual steps and present further important contributions to the analysis of the very notion of computability.

2.2 Puzzles vs. Calculable Functions

Hilbert had asked about the "mechanical decidability" of mathematics, while Turing proved its undecidability by processes his machines can carry out. So why are Turing machines not regarded as ad hoc artifacts introduced by Turing for the sole purpose of giving a cheap, negative answer to Hilbert's problem? Why is Turing's

undecidability theorem regarded as a fully satisfactory solution to the *Entscheidungsproblem*?

To partially answer this central question in the current section, we will not start out with Turing's 1936/37 paper, but rather begin with his paper *Solvable and unsolvable problems* (Turing 1954) in which Turing formulated an "enigmatic variant" of his thesis. This paper, presumably Turing's last publication, appeared in 1954 in the journal *Science News*; it focuses on the methodological problems surrounding decidability and argues for the adequacy of a rigorous notion of computability *without* ever mentioning his machines. That may be surprising; after all, for Church, Gödel, and many contemporary computer scientists, it is the analysis of mechanical procedures in terms of "finite machines" that makes Turing's work so convincing. Instead of "machines", Turing uses "unambiguous substitution puzzles" as the basic concept in this paper.

2.2.1 Substitution Puzzles

The *enigmatic variant of Turing's Thesis* is articulated as follows: given any puzzle, we can find a "corresponding substitution puzzle, which is equivalent to it in the sense that given a solution of the one we can easily find a solution of the other." This statement that any puzzle has a substitution puzzle as its *normal form* may be puzzling in itself, and even Turing had to admit that it is "somewhat lacking in definiteness, and will remain so". He views this not-quite-definite statement as being situated between a theorem and a definition:

> In so far as we know *a priori* what is a puzzle and what is not, the statement is a theorem. In so far as we do not know what puzzles are, the statement is a definition that tells us something about what they are. (Turing 1954, p. 15)

Let us look at some of Turing's puzzles. He considers as paradigmatic examples the 2-dimensional sliding squares and the 3-dimensional knots. The knot problem is like a puzzle in that it asks "to undo a tangle, or more generally of trying to turn one knot into another without cutting the string". He then introduces 1-dimensional substitution puzzles. To play them, one has an unlimited supply of counters, possibly only of two distinct kinds, namely black (B) and white (W). An initial configuration is a finite sequence of such counters, and the puzzle task is to transform the given configuration into another one by substitutions from a fixed finite list; these substitutions specify the admissible moves in the puzzle. As an example Turing considers the substitutions (i) WBW → B and (ii) BW → WBBW and shows how they can be used to transform WBW into WBBBW. He also remarks that, for instance, WBB cannot be transformed into BW, "for there are no admissible steps which reduce the number of B's".

Another good, and not just playful, example of a substitution puzzle is "the task of proving a mathematical theorem within an axiomatic system". This is nothing but Hilbert's *Entscheidungsproblem*, viewed as one of the most important issues in mathematical logic during the 1920s. As we already saw, Turing had addressed in

his (Turing 1936/37) the key conceptual problem of explicating "mechanical procedures" by his computing machines and showed the unsolvability of the decision problem by "reducing" the printing or halting problem to it.[1] He used this method of reduction also in his (Turing 1950) in order to establish the unsolvability of another problem; however, he remarked at the beginning of the paper:

> 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 ([1936]) and the author ([Turing 1936/37]). (p. 491)

Turing points through the last part of his remark, in a striking way, to the structural similarity between his two-letter machine and the *worker* in (Post 1936). *Post's worker* operates in a *symbol space* consisting of a two-way potentially infinite sequence of marked and unmarked boxes. The worker can perform a number of *primitive acts*: mark or unmark a box, move to the box immediately to the right or to the left of the box he is in, and determine whether the box he is in is marked or not. When carrying out a procedure the worker follows a finite, numbered sequence of instructions. Post gave no intrinsic reasons for this formulation, but in the background is his extensive work from the 1920s on *production systems* and their *normal forms*. The mathematical core of that work was published only much later in (Post 1943).

Post described in his (1947) Turing machines *via* production systems and proved the word problem for semigroups to be unsolvable by reducing the halting problem to it. Three years later, Turing used Post's general approach to extend the unsolvability result from semigroups to semigroups with cancellation. This concrete mathematical work is part of the reason why Turing in (1954) exposes his fundamental considerations on computability via substitution puzzles. However, the main reason for his expository choice seems to have been the *conceptual confluence* of his work with Post's, to be discussed in Sect. 2.3.1. For a fuller exposition see (Davis and Sieg 2015).

2.2.2 Calculable Functions

Turing's earlier remarks, quoted at the very beginning of the last subsection, give a hint of his perspective on the methodological status of his *Thesis*. The thoughts of other mathematical logicians like Hilbert, Bernays, Herbrand, Gödel, … were not at all proceeding along those lines, but were informed by paradigmatic *primitive recursive procedures* for mechanically calculating the values of number theoretic functions. Examples of such procedures for natural numbers were familiar from mathematical practice. Indeed, most functions from elementary number theory are in this class, as was established already by Skolem (1923). Among them are addition, multiplication, exponentiation, the factorial function $n!$, the characteristic function of

[1] Note that in Turing's remark below this contemporary understanding of reduction is "reversed": Turing's "reducing the unsolvability of A to the unsolvability of" amounts to "reducing the problem B to the problem A".

the set of prime numbers, and the Euler function φ. Nevertheless, Hilbert's student Ackermann (1928) discovered in the mid-1920s a non-primitive recursive function whose values could be determined by an obviously mechanical procedure. (Schemes for non-primitive recursive functions whose values are calculable were systematically investigated by Péter in the mid-1930s; see her book *Recursive Functions* (1967)).

Dedekind had introduced primitive recursive functions in his (1888). Hilbert and Bernays used them in the 1920s with a deeply computational perspective. Herbrand in his (1931) expanded the class and Gödel built on that work when introducing the even broader class of *general recursive functions* in (1934). Gödel considered it as an "important property" that the value of a primitive recursive function can be computed by a "finite procedure", for any argument; and he added in a footnote:

> The converse seems to be true if, besides recursions according to the scheme (2) [that of primitive recursion], recursions of other forms ... are admitted. This cannot be proved, since the notion of finite computation is not defined, but it can serve as a heuristic principle.

The general recursive functions are defined in the last section of Gödel's Princeton lectures as unique solutions of functional equations, and — what was crucial for Gödel — their values are uniformly computable from generalized recursion equations by clearly mechanical rules. Gödel presented in his (193?) a simplified form of his equational calculus with just two rules, namely, (**R1**) that allows substituting numerals for variables, and (**R2**) that allows substituting equals for equals (i.e., if the equation $T_1 = T_2$ has been obtained for terms T_1 and T_2, then occurrences of T_1 can be replaced by T_2 in other provable equations).

2.2.3 Church's Thesis

Church used Gödel's notion of a general recursive function in the first published formulation of his *Thesis* in (1935): the calculable number theoretic functions are to be *identified* with the general recursive ones. Church argued for this identification by appealing to quasi-empirical evidence (all known effectively calculable functions are general recursive) and by pointing to the provable equivalence of different notions (for example, of general recursiveness and λ-definability). The second reason is still a very central part of the standard presentation of Church's Thesis and proceeds as follows. The work of Gödel, Church, Turing, and others (e.g., Kleene, Post, Hilbert, Bernays) provides mathematical definitions of mechanical procedures. These different and differently motivated definitions characterize the same class of number theoretic functions, and this remarkable *confluence* is considered as strong evidence for the correctness of Church's Thesis. (See for example (Gandy 1988).) For us, the question is rather: What are the grounds for accepting the various notions as constituting a mathematical characterization of mechanical procedures?

In an attempt to answer this question, we focus on arguments that analyze calculability of number theoretic functions in terms of one central concept, namely, *calculation in a logic*. Church, in his classical paper (1936), points out that the notion of

calculability in a logic suggests itself as a way to explicate effective calculability, and he argues that it does not lead to a definition more general than recursiveness. Let us indicate the argument: Church considers a logic *L*, i.e., a system of symbolic logic whose language contains the equality symbol =, a symbol { }( ) for the application of a unary function symbol to its argument, and numerals for the positive integers. A unary function F is called *effectively calculable* if there is an expression *f* in *L* such that {*f*}(M)=N is a theorem of *L* exactly when F(*m*)=*n*; M and N are expressions of *L* that stand for the positive integers *m* and *n*. Church claims that any such F is general recursive, when *L* satisfies conditions that guarantee essentially the recursive enumerability of *L*'s theorem predicate; the claim follows by an unbounded search. The crucial condition in Church's list requires the *steps* in derivations of equations to be, well, general recursive! Here we hit a serious stumbling block for Church's analysis, since it is logically circular to appeal to the thesis when arguing for it. And yet, Church's argument achieves something: the general concept of calculability is explicated as derivability in a symbolic logic, and the step-condition is used to sharpen the idea that we operate by effective rules in such a formalism. We suggest calling the claim "Steps of any effective procedure must be recursive!" *Church's Central Thesis*.

Church's concept of calculability in a logic generalizes Gödel's notion of calculability in a restricted equational calculus; it is extremely natural and fruitful. It is also directly related to decidability (*Entscheidungsdefinitheit*) for relations and classes in Gödel's 1931 paper and to *representability* in his 1934 Princeton lectures. Gödel defined the very notion for higher-order logics in his 1936 note "On the length of proofs" and emphasized its *type-absoluteness*. Finally, in his contribution to the Princeton Bicentennial Conference of 1946, Gödel took absoluteness (in an even more general sense) as the main reason for the special importance of recursiveness. Here we have, according to Gödel, the first interesting epistemological notion whose definition is not dependent on the chosen formalism. But the stumbling block Church had to face shows up also here: absoluteness is achieved only relative to *formal* systems. (For a detailed analysis of Gödel's considerations, see Sieg's paper *Gödel on Computability* (2006).)

The notion of absoluteness Gödel gave in 1946 is implicit in Hilbert and Bernays' *Grundlagen der Mathematik II* (1939/1970). They called a number-theoretic function *reckonable according to rules* (*regelrecht auswertbar*), if it is calculable in some deductive formalism, and formulated three recursiveness conditions for such formalisms. Then they proved: (i) a function that is reckonable in some deductive formalism satisfying the recursiveness conditions can be computed in a very restricted number theoretic formalism, and (ii) the functions calculable in the latter formalism are exactly the recursive functions. Hilbert and Bernays' analysis is a most satisfactory capping of the development from *Entscheidungsdefinitheit* to an absolute notion of computability. But their analysis does not overcome the major stumbling block; rather, it puts the stumbling block in plain view through the recursiveness conditions that deductive formalisms must satisfy. The crucial condition requires the proof predicate for such formalisms to be primitive recursive, without arguing for that requirement.

2.3 Mechanical Procedures

What we described in Sect. 2.2.3 are the most sophisticated arguments for Church's Thesis we know of. They are important, but provide neither a proof nor convincing reasons; after all, function values are being *calculated in a logic via* elementary steps, and these elementary steps are taken to be (primitive) recursive. When looking back at the mathematical problems and the conceptual issues that had to be addressed, one notices that they are hardly concerned with such calculations; they also seem to have nothing in common with operations of a Turing machine or the wanderings of a Post worker. Well, let's examine the last observation by reviewing Turing's 1936/37 analysis of calculability and by pointing to its distinctive feature that leads to fundamental boundedness and locality conditions. These conditions are then incorporated into an abstract, axiomatic characterization of a *computable process*.

2.3.1 Turing's Analysis

Turing emphasizes in his (1936/37) from the very beginning that he is concerned with *mechanical operations* that are carried out by humans *on symbolic configurations*. He uses the term *computer* for a human computing agent who proceeds mechanically. Gandy (1988) suggested calling computers in Turing's sense *computors* and computing machines *computers*. Initially, the symbolic configurations on which computors operate are written on paper that is divided into squares "like a child's arithmetic book". The two-dimensional character of the paper, however, is not viewed as essential to computations, and the one-dimensional tape divided into squares is taken as the basic computing space.

Turing strives then to isolate computor-operations that are "so elementary that it is not easy to imagine them further divided" and formulates a crucial requirement: symbolic configurations relevant for a computor's actions have to be recognized *immediately or at a glance*. Because of the step from the two-dimensional to the one-dimensional computing space, one has to be concerned only with finite sequences of symbols and their immediate recognizability. Now, Turing appeals to a crude empirical fact, namely, that it is impossible for a computor to determine at a glance whether or not the sequence 11127778199999666633452 is identical with 11127778199999666633452. The computor's sensory limitation leads directly to boundedness and locality conditions: (**B**) the number of symbol sequences a computor can recognize at a glance are uniformly bounded, and (**L**) the operations a computor can carry out modify locally a recognized configuration.

Given this restrictive analysis of a computor's steps, it is evident that Turing machines operating on strings (*string machines*) simulate computors. In order to finally connect Turing with Post, we notice that string machines are substitution puzzles, that the latter are a species of Post's production systems, and that the substitution puzzles subsume all Turing machines. To clarify this final claim, we have to

mention that *letter machines* can simulate string machines. A letter (Turing) machine consists of a finite, but potentially infinite tape; the tape is divided into squares, and each square may carry a symbol from a finite alphabet, say, just the two-letter alphabet consisting of 0 and 1. The machine can scan one square at a time and, depending on the content of the observed square and its internal state, it prints 0 or 1, but shifts also attention to one of the two immediately adjacent squares and changes its internal state. This is done in accord with the machine's program, a predetermined finite list. The machine's operation is deterministic in the following sense: the scanned square and the current internal state determine uniquely the machine's action and next internal state. Coming back to the clarification we intended to achieve, the computations of a letter machine can be characterized by a substitution puzzle. That completes the argument for the conceptual confluence of Turing's analysis and Post's work.

Turing viewed the above reductive argument as mathematically unsatisfactory because it relies on an appeal to "intuition". However, an appeal to intuition is no longer needed for the *full Thesis*, stating that every calculation of a computor on general symbolic configurations can be carried out by a suitable Turing machine; rather, it suffices to claim that the computor's calculations can be carried out on strings, without loss of generality. Let us call this judgment *Turing's Central Thesis*. As we did in the reflections concerning the calculability of number theoretic functions, here we are also connecting an informal notion (*symbolic configuration*) with a precise mathematical one (*string*). Allowing larger classes of mathematically precise configurations makes Turing's Central Thesis inductively more convincing. For example, Kolmogorov & Uspenski considered in (1958) particular kinds of graphs, and Sieg and Byrnes generalized them further to K-graphs in their (1996). However, an appeal to an analogue of Turing's Central Thesis can't be avoided. That may have been the reason why Turing thought that the enigmatic variant of his Thesis (formulated in his (1954) and discussed at the beginning of Sect. 2.1 above) must remain indefinite. Thus, he considered the statement as one "which one does not attempt to prove". We agree, but we also think one can do better than resort to "propaganda": That's what Turing suggested, tongue in cheek; after all, the status of this statement "is something between a theorem and a definition".

The better way starts out with a direct observation contrasting calculations in Gödel's equational calculus via the rules (**R1**) and (**R2**) with those a computor can carry out, for example, on strings or K-graphs. (**R1**) permits us to replace in a single step a variable by an arbitrary numeral; thus a single symbol can be replaced by immense strings that constitute numerals. The prima facie unboundedness in this calculation step is dramatically restricted in the case of Turing computations: not only is the number of immediately recognizable configurations uniformly bounded by (**B**), but so is, on account of (**L**), the number of possible different replacement steps. This contrast applies equally when one considers calculations in a logic according to Church or in a deductive formalism according to Hilbert and Bernays. In the next subsection we take advantage of this particular feature of Turing computations and outline an "axiomatic approach" to computability – by *defining an abstract notion of computability through axioms*, thus joining what Turing saw in (1954) as disjointed. That of course has to be explained!

2.3.2 Abstract Concepts

The new way involves, to put it differently, the introduction of a more abstract concept of computable processes. Analogous steps were already taken in nineteenth century mathematics. There is an abundance of *structural definitions* in modern mathematics: groups, fields, topological spaces; they reflect the move in mathematics to greater abstractness. Let us see where we stand with respect to an abstract conception of *mechanical processes*. In the last subsection we generalized the configurations in "Post production systems" from strings to K-graphs. Let $\underline{S}$ be the set of strings (of elements from some finite alphabet); a function f from $\underline{S}$ to $\underline{S}$ is called $\underline{S}$-*computable* if there is a string machine such that the value of f, for any argument, is determined by a computation of that machine. Recall that string machines are substitution puzzles, and that this allows us to see that replacing strings by K-graphs leads to an immediate generalization. So let $\underline{K}$ be the set of K-graphs (over some finite alphabet); a function ψ from $\underline{K}$ to $\underline{K}$ is called K-*computable* if there are finitely many $\underline{K}$-transition rules such that the value of the function ψ, for any argument, is determined by a computation following the rules. The detailed considerations in (Sieg and Byrnes 1996) can be adapted to prove that any $\underline{K}$ -computable function ψ is $\underline{S}$-computable (under a suitable encoding of K-graphs as strings). For the other direction just notice that $\underline{S}$-transition rules specify string machines. Thus, these two notions turn out to be equivalent.

The real question is, however: Can we get away from treating special, rigorously specified kinds of configurations? As a step towards that aim, let us reformulate the informal *requirements for computors*: (i) they operate on finite, but unbounded configurations; (ii) they can recognize, in each configuration, a unique pattern of a fixed, bounded list; (iii) they locally operate on the pattern recognized in a given configuration; (iv) they assemble the next configuration from the original one and the result of the local operation. These requirements can be formulated as *characteristic condition*s for *computable* discrete dynamical systems; see Chapter VI of (Jacobs 1992) for a basic introduction to such systems. Consider a discrete dynamical system with state-space $\mathscr{S}$, consisting of hereditarily finite sets, and an operation $\mathscr{F}$ from $\mathscr{S}$ to $\mathscr{S}$. Such a dynamical system is by definition *computable* if the operation $\mathscr{F}$ is computable. In turn, the operation $\mathscr{F}$ is called *computable* if there are finitely many patterns and local operations on them such that the value of $\mathscr{F}$, for any argument, is determined by a computation via the local operations. With this definition in place it is not difficult to see that Turing machines can be viewed as computable discrete dynamical systems. More importantly, one can prove a *Representation Theorem* that states: The computations of any discrete dynamical system satisfying the characteristic computability conditions can be reduced to those of Turing machines.

The closest analogy to this motivated axiomatic introduction of an abstract concept is the move from particular "constructions" of real numbers, for example, as Dedekind cuts or Cauchy sequences, to an abstract notion of a complete ordered field that was then seen to be categorical. Nevertheless, we may have left the reader

with a puzzle in the ordinary sense: How are these considerations to be made mathematically precise? Details are found in Sieg's paper "On Computability" (2009), where an axiomatically characterized abstract notion of a *Turing Computor* is introduced; that notion corresponds to what we called here a computable discrete dynamical system. The guiding ideas that led to the concept of such a computor system can be used to obtain also an analysis of *machines* and of *parallel machines*.[2] Gandy proposed in his (1980) such an analysis and used, as Turing did, a *Central Thesis* to connect parallel computations satisfying informal restrictive conditions to particular kinds of dynamical systems. In Sieg's "On Computability" (2009), Gandy's approach is thoroughly reworked to an explicit axiomatic analysis of machine computability appealing only to restrictive physical conditions. The latter are motivated by two physical facts: (i) a lower bound on the size of physically distinguishable "atomic" components justified by the uncertainty principle of quantum mechanics, and (ii) an upper bound on signal propagation grounded in the theory of special relativity. Together, the restrictions (i) and (ii) form the basis of boundedness and locality conditions for machines in the way sensory limitations do for computors.

We have to come back now to the beginning of our paper, when we discussed two distinctive features of Turing machines. Up to now we have only discussed the first, expressing that any algorithm that can be carried out by a computor can be computed by some Turing machine. The second feature is not only of great theoretical interest, but also of enormous practical significance, as it has been adopted for the architecture of our contemporary digital computing machines. This feature expresses that a single Turing machine can simulate any Turing machine on any input; such a machine is called *universal*.

2.4 The Universal Turing Machine

A *universal machine* was defined by Turing as a mathematical object U, like Euler's number $e = 2.7182818284\ldots$. In contrast to e, U is nothing but a carefully crafted finite list of symbols that lends itself to materialization into a versatile and powerful object, our computer. In fact, the latter computes (within some practical limits) like U all the functions Turing machines compute.

[2] Machines are here taken to be real physical mechanisms that consist of finitely many parts and operate in discrete steps; Gandy called them *discrete mechanical devices* and contrasted them to analogue machines. Parallel machines locally operate on parts of a given configuration; the results of the local operations are then joined into the next configuration. The paper (Mundici 1983) discusses Turing machines as physical objects under the effect of special relativity. Further papers by Mundici, quoted in this paper, discuss also quantum mechanical limitations. None of these papers, however, aims at motivating the boundedness and locality conditions for (Turing or parallel) machines.

2.4.1 The Digitization of Turing Machines

For the construction of a machine U that is able to simulate (= do the same computations as) every Turing machine T, one cannot use the infinite alphabet given by the union $\{a_1, a_2, ...\}$ of all alphabets of all possible Turing machines. A cheap trick solves this problem: write each symbol a_t in "unary notation" as the symbol sequence a | |...| , i.e., the letter a followed by t strokes; we call this transcription the *digitization* of a_t. In particular, a | digitizes the blank symbol of the tape of T, whereas the symbol □ denotes the blank symbol of the tape of U. Similarly, we digitize all possible states of Turing machines that are indicated by natural numbers 1, 2, 3, … by s |, s | |, s | | |, ...

We are now ready to digitize any given Turing machine T. The name "machine" suggests hardware, but remember that T is just a finite list of quintuples of the form $(r, a_i, a_j, \Leftarrow, t)$ or $(r, a_i, a_j, \Rightarrow, t)$, where r and t are natural numbers indicating states, and a_i and a_j are symbols of the alphabet of T. We read, e.g., $(r, a_i, a_j, \Leftarrow, t)$ as the instruction: "if in state r you read on the scanned square the symbol a_i then replace it by a_j move to the square immediately to the left of the scanned square, and change to state t". As a finite list of such quintuples, T is something purely symbolic; it is a program, i.e., software. Thus, the finite alphabet A={a, s, |, ⇒, ⇐ }, fixed once and for all, suffices to digitize the instructions of T. For instance, the instruction $(3, a_2, a_1, \Rightarrow, 1)$ is digitized as s | | | a | | a | ⇒ s |. The machine T is now digitized as the sequence of its digitized instructions, one next to the other, without commas or separation symbols between any two consecutive instructions. Summing up, this digitization process transforms every Turing machine T into a string $\underline{T}$ over the alphabet A.

Suppose we are given a Turing machine T with its alphabet $A_T = \{a_1, a_2, ... , a_k\}$, where a_1 is the blank symbol of T's tape, and its set of states $S_T = \{m_1, m_2, ... , m_k\}$, where each m_i is a natural number. A *configuration* C of T has the form

$$\begin{array}{c} r \\ \downarrow \\ a_{n(1)} \ldots a_{n(t)} a_{n(t+1)} \ldots a_{n(u)}; \end{array}$$

C expresses that T's tape content is given by the sequence $a_{n(1)} ... a_{n(t)} a_{n(t+1)} ... a_{n(u)}$, that T is in state r reading the square containing the symbol $a_{n(t+1)}$. The configuration C is all we need to define the next configuration D and the computation step (C,D). The corresponding *digitized configuration* $\underline{C}$ of C is obtained by placing the digitized state s | | | |...| (with r strokes) immediately to the left of the symbol $a_{n(t+1)}$ scanned by T, and by digitizing all the symbols of C. Letting $|_n$ be short for | |...| (n strokes), the result is the following string of symbols of A:

$$a|_{n(1)} \ldots a|_{n(t)} s|_r a|_{n(t+1)} \ldots a|_{n(u)}$$

This digitization is possible because, by definition, only finitely many squares of T's tape are nonblank. The proviso that all squares outside the above picture are blank is tacitly understood; those squares play obviously no role in determining the configuration that succeeds C in a computation. Implicitly, we have made use of one dramatic simplification introduced by Turing himself. Recall, Turing analyzed a computor's computations and made steps in the computation dependent on the computor's "state of mind". The appeal to such mental states is a little later in (1936/37) avoided as follows:

> It is always possible for the computer [i.e., in our terminology, the computor] to break off from his work, to go away and forget all about it, and later to come back and go on with it. If he does this he must leave a note of instructions (written in some standard form) explaining how the work is to be continued. This note is the counterpart of the "state of mind". We will suppose that the computer works in such a desultory manner that he never does more than one step at a sitting. The note of instructions must enable him to carry out one step and write the next note.

Turing thus replaces "mental states" by "physical counterparts" that can then be built into the configurations of machines as we did!

2.4.2 Construction of the Universal Machine

Given a Turing machine T and two configurations C and D we will write T: $C \rightarrow_1 D$ to mean that T produces D as the configuration succeeding C. In other words, the pair (C,D) is a *computation step* of T. We will write T: $C \rightarrow D$ to mean that T, starting with C, produces D after a finite number of such steps; i.e., there is a finite list of configurations $C_0, C_1, \ldots, C_v$ such that $C_0=C$, $C_v=D$, and for each $i=0,\ldots,v-1$ we have T: $C_i \rightarrow_1 C_{i+1}$.

Theorem (Turing 1936/37) One can construct a Turing machine U_1 such that, for every Turing machine T and every pair (C,D) of configurations of T, the following holds:

- If T: $C \rightarrow_1 D$ then U_1: $\underline{T} \square \underline{C} \rightarrow \underline{T} \square \underline{D}$. In greater detail: upon placing U_1 in its initial state i over the first symbol of the digitized input $\underline{T} \square \underline{C}$, U_1 terminates after finitely many steps in a final state h on the initial symbol of the output string $\underline{T} \square \underline{D}$.
- If T halts over configuration C (for want of instructions), then upon placing U_1 in its initial state i over the initial symbol of the digitized input $\underline{T} \square \underline{C}$, U_1 terminates after finitely many steps in state h on the blank symbol to the left of the input string $\underline{T} \square \underline{C}$.

Proof (Sketch) As above, we suppose that the configuration $\underline{C}$ is given by

$$a\Big|_{n(1)} \cdots\cdots a\Big|_{n(t)} s\Big|_{r} a\Big|_{n(t+1)} \cdots a\Big|_{n(u)}$$

While $\underline{C}$ may consist of millions of symbols of the alphabet A, only its fragment $a\ |_{n(t)}\ s\ |_r\ a\ |_{n(t+1)}$ will be changed in the step leading to $\underline{D}$. In case the (necessarily unique) instruction $\underline{I}$ of $\underline{T}$ beginning with $s\ |_r\ a\ |_{n(t+1)}$ is right-handed, for some natural numbers k and h, $\underline{I}$ will have the form $s|_r\ a\ |_{n(t+1)}\ a\ |_k \Rightarrow s\ |_h$, and $a\ |_{n(t)}\ s\ |_r\ a\ |_{n(t+1)}$ will be replaced by $a\ |_{n(t)}\ a\ |_k\ s\ |_h$. In case $\underline{I}$ is left-handed, for some natural numbers i and j it will have the form $s\ |_r\ a\ |_{n(t+1)}\ a\ |_i \Leftarrow s\ |_j$ and the new fragment will be $s\ |_j\ a\ |_{n(t)}\ a\ |_i$. Needless to say, if no instruction of $\underline{T}$ begins with $s\ |_r\ a\ |_{n(t+1)}$ then $\underline{C}$ has no succeeding configuration $\underline{D}$.

The operations of selecting $\underline{I}$ among the instructions of $\underline{T}$, (or else, verifying that no such $\underline{I}$ exists), and carrying out such $\underline{I}$, involve very simple "routines" or "modules" well within the reach of Turing machines, such as:

- highlight a piece P of text: for this purpose U_1 is equipped at the very outset with a copy A´ of the alphabet A, where A´ contains the boldface version of the symbols of A; then highlighting P means rewriting P in boldface as a piece of text P´,
- un-highlight P´ to P,
- cancel, move or duplicate highlighted pieces of text,
- check whether two pieces of text are identical symbol by symbol,
- look for pieces of the tape containing a string that begins with a given substring
- place side by side two pieces of non-blank text, deleting the blank interval that separates them.

At the moment we have a machine U_1 that simulates a single computation step of an arbitrary Turing machine T. The Corollary joins the single step simulations into a simulation of T's computations.

Corollary One can construct a Turing machine U such that for every Turing machine T the following holds: for every pair of configurations C and E with T: C → E, if T halts on E and if we place U in its initial state i over the first symbol of the digitized input $\underline{T}$ □ $\underline{C}$, then U halts after a finite number of steps on the first symbol of $\underline{T}$ □ $\underline{E}$. If T does not halt on input C, then U does not halt on input $\underline{T}$ □ $\underline{C}$.

Proof It is sufficient that the machine U_1 of the Theorem, once it has reached its final state h, rather than stop, passes the control to its own initial state i. This is trivially achieved by replacing every instruction of U_1 of the form (r, a, b, ⇒, h), respectively (r, a, b, ⇐, h) by the instruction (r, a, b, ⇒, i), respectively (r, a, b, ⇐, i).

Having constructed the universal machine that is so crucial for the architecture of digital computers, thus for computational practice, let us return to the starting point of our considerations – that are remarkably theoretical and, some might say, esoteric: In Sect. 2.1 we gave an account of Turing's argument to settle the *Entscheidungsproblem* in the negative. Sects. 2.2 and 2.3 have provided enough material to substantiate Turing's claim that the undecidability of first-order logic by Turing machines amounts to the actual unsolvability of Hilbert's imprecisely stated *Entscheidungsproblem*. In the current section, we have presented an added bonus of his machines, the all-important universal Turing machine. In order to make his theorem independent of the formal features of PM or any other concrete "formal" system like ZF, Gödel needed a general concept of a "formal system". Gödel showed

in his (1931) that all syntactic notions for the description of PM are primitive recursive in their arithmetized form. He introduced in his 1934 Princeton Lectures, however, the class of general recursive functions as discussed in Sect. 2.2. At the time, he did not think that this broader class of effectively calculable functions captured the informal concept of calculability. (As to the developments of Gödel's view, see (Sieg 2006).) In the Postscripts to his (1931) and (1934) that were written in 1963 and 1964, respectively, Gödel most strongly asserted that Turing's definition fully captures the informal concept of calculability and that it provides "an unquestionably adequate definition of the general concept of formal system". Using this concept he extended the incompleteness theorems to *all* sufficiently strong formal theories.

2.5 Polynomial Time

Turing computability is a highly idealized concept as it disregards limitations on at least two resources, namely, space and time. The former is disregarded, as the number of tape squares scanned during a computation is unbounded; the latter is disregarded, as the number of computation steps is not limited. If we insist that the number of steps be bounded, we automatically insist on a bound for the number of tape squares that can be used in a computation. Yet it seems necessary to impose bounds: the size of the physical universe is bounded, and so is our lifespan. Intuitively speaking, a problem is *tractable* if it can be solved "quickly", possibly via some instructive shortcut argument using new and sophisticated concepts, such as symmetry, counting, induction, or continuity. Such concepts may take care of an enormous number of elementary subcases, just as a single multiplication takes care of many additions.

To grasp the need for shortcuts, suppose we are in a highly inflationary country, whose currency is named "flip". The random generator of a computer has listed a set X of 300 positive integers n_i in the range R from 1 to 10^9. These integers are viewed as values of stamps, and we are asked to check whether a postage Σ of 150 x 10^9 flips can be exactly achieved by suitably choosing a subset of the stamps in X. The answer is likely to be negative, but who would set out to check all possible 2^{300} subsets of X to confirm this? And yet, if some clairvoyant were to *guess* such a subset Y, then we could happily *check* that his guess was right — by just verifying that the values of the stamps in Y add up to the required postage. This pair (X, Σ) is an instance of an innocent looking problem, known as KNAPSACK. Already for small sets of stamps and small postages, instances of this problem cannot even be solved by government pundits with their latest supercomputers. As of today, KNAPSACK is an intractable problem.

To better appreciate the distinction between guessing (the clairvoyant's astral job) and checking (our pedestrian job), recall Fermat's conjecture that for each *n* the number

$$1+2^{(2^n)}$$

is prime. He had checked the truth of the conjecture for n=0,1,2,3,4. For n=5, however, Euler (magically?) obtained the factorization 641 x 6700417: any schoolboy can check as a routine multiplication exercise that this product equals 4294967297, which in turn equals $1+2^{(2^5)}$. The problem of finding a subset Y of a set X of numbers so as to realize a total sum Σ is not "tractable". It would become tractable with the help of a clairvoyant guessing, correctly, appropriate subsets Y of X. Can the notion of tractability be given a mathematically precise sense in the same way as it was done for the general notion of computability?

In a letter to von Neumann of 20 March 1956 (2003, pp. 373–77), Gödel considered a Turing machine T that decides, for any formula F of the predicate calculus and any natural number n, whether F has a proof of length n. Let TIME(F,n) be the number of steps T requires to decide, whether F has such a proof; let MAXTIME(n) be the maximum of TIME(F,n) as F ranges over all formulas (necessarily of length ≤ n). Gödel asked von Neumann to confirm his belief that MAXTIME(n) is a slow-growing function, perhaps growing proportionally to n or to the square of n. Gödel noted that, if his belief were true, computers could replace human reasoning for such yes-no problems - in spite of the unsolvability of the *Entscheidungsproblem*. Gödel asked von Neumann also about the possibility of fast procedures deciding questions such as whether a given number is prime. (This was answered positively by Agrawal et al. (2004)). We don't know whether von Neumann answered Gödel's letter. Von Neumann was seriously ill at the time and died on 8 February 1957. In any event, Gödel adumbrated a quantitative measure for the complexity of solutions to decidable problems and, thus, a measure for their tractability. (The correspondence between Gödel and von Neumann is found in Gödel's *Collected Works V* (Gödel 2003).)

The step from decidability to tractability is the birth of complexity theory: Instead of asking whether a problem is algorithmically solvable, attention is focused on the resources needed for a solution. Today we know that MAXTIME(n) is bounded by a polynomial if and only if P=NP. Gödel's optimism on the growth of MAXTIME(n) was not shared by the Russian cybernetic school: That school investigated, around 1956, the role of exhaustive search ("*perebor*") and concluded that for several important problems perebor seemed inevitable. However, a formal definition of a "universal perebor problem" was introduced by Levin only in his (Levin 1973), almost at the time when Cook proved his famous theorem. (As to the latter theorem, see Sect. 2.6.1.)

To give a mathematical definition of "easy" and "difficult" problems, we fix a finite set A and call it our *alphabet*. Elements of A will be called *symbols* of A, and we denote by A* the set of all words, i.e., all finite strings of symbols of A. By a *problem* we mean a subset of A*. Thus for instance, letting A={0,1, ... ,9}, consider the set PRIME={2,3,5,7,11, ...} of all prime numbers; PRIME is identified with the "problem" of deciding whether a number is prime. Similarly, the problem of recognizing COMPOSITE numbers is by definition identified with the set

COMPOSITE={4, 6, 8, 9, 10, 12, ...}. Given some fixed alphabet to write down formulas in propositional logic, the satisfiability problem SAT is the set of all formulas with a satisfying truth value assignment. This *extensional* identification of problems with sets of words is a preliminary, but necessary step to define *intensional* notions for the complexity of problems. We measure the complexity of a problem L in terms of the number of Turing steps needed to decide whether an arbitrary word of A* does or does not belong to L. Specifically, we say that a problem L is *decidable in polynomial time* (in *polytime*) or more briefly L *is in* P, if there is a polynomial r and a Turing machine T with the following property: Having as its input an arbitrary word x of A* with n (occurrences of) symbols, T decides x in L within at most r(n) steps.

The reader who has consulted a few textbooks on Turing machines may have observed that there is no unanimity about the details of Turing machines. Here are a few differences: some books use quintuples (as we did), others use quadruples; some use tapes that are infinite in both directions (as we did), others use right-infinite tapes; some use not only single tapes, but double tapes. Fortunately, a simple simulation argument shows that if a problem is solvable in polynomial time by machines in one model, then it is also solvable in polynomial time by machines in any other reasonable model. Technically speaking, the class P is "robust" or "textbook-invariant".

Let us mention a little bit of the history of these developments. In his (1960), Rabin made explicit for the first time the notion of computational complexity for functions. Cobham noted in his (1964) that many functions can be computed in polynomial time and that the notion of polynomial time computability is robust. Edmonds, in his (1965), called an algorithm good whose complexity grows polynomially with the length of the input.

2.6 NP-Completeness

There is a very long list of problems, coming from all fields where computation is done, for which no polynomial time algorithm is known. Many examples are found in the classical text (Garey and Johnson 1979). The list includes KNAPSACK, COLORABILITY (the problem of coloring the vertices of a graph so that any two vertices connected by an arc have different colors from a palette with k colors), CLIQUE (the problem whether a graph H has k pairwise connected vertices), INDEPENDENT SET (the problem whether a graph H has k pairwise disconnected vertices), SAT. A closer examination shows that for every formula F, the task of deciding whether F is in SAT can be split into two sub-tasks:

(i) magically guessing a satisfying assignment for F,
(ii) trivially checking that the guessed assignment satisfies F.

Similarly, for KNAPSACK one must guess-and-then-check a subset Y of X achieving a postage Σ; for COLORABILITY a coloring; for CLIQUE a set of pair-

wise connected vertices; for COMPOSITE a nontrivial divisor... . Edmonds called a problem *well-characterizable* if every solution has a polynomial time checkable certificate.

To enable a Turing machine T to perform subtask (i), we allow T to choose non-deterministically among finitely many states and actions. While in the case of a deterministic Turing machine, given input x, the possible configurations of T can be listed one after another, in the non-deterministic case these configurations are conveniently located in a (genealogical) tree: The initial configuration is at the top of the tree; at each branching point there is a configuration C having finitely many possible immediate successors C′, C″, ... as given by the quintuples of T taking effect in C. T *accepts* x if the tree has a path of successive configurations ending with a halting configuration.

We are now in a position to formalize the guessing/checking procedure (i)-(ii). We say that a problem L *is in* NP if there is a polynomial r and a non-deterministic Turing machine T such that for any n and string x of length n, T has an accepting computation of less than r(n) steps, if x is in L, and T has no accepting computation at all, if x is not in L. It follows from the definition that SAT, COLORABILITY, CLIQUE, INDEPENDENT SET, KNAPSACK, and COMPOSITE are all in NP. While COMPOSITE has been proved to be in P, none of the other mentioned problems has so far been proved to be in P.

2.6.1 Reductions and NP-Complete Problems

We now introduce the appropriate dynamic machinery that enables us to pass from one problem to another. This will be a variant of Turing's reduction of the halting problem to the tautology problem in first-order logic. Suppose we need a machine A for the CLIQUE problem, but we only have a machine B for the INDEPENDENT SET problem. Thus over input (G,k) with G a graph and k a positive integer, machine B decides whether G has k pairwise disconnected vertices. We can efficiently use B for the CLIQUE problem, i.e., the problem whether a graph H has k pairwise connected vertices: For this purpose it is sufficient to modify H into its dual graph H', whose arcs are precisely the arcs missing in H. Then the machine for INDEPENDENT SET accepts (H',k) if and only if H has k pairwise connected arcs. The transformation from (H,k) to (H',k) quickly reduces CLIQUE to INDEPENDENT SET.

The following definition is due to Karp (1972): A subset L of A* (i.e., a problem L over alphabet A) is *polynomially reducible* to a subset M of B* (i.e., a problem M over alphabet B) if there exists a polynomial r and a deterministic Turing machine T such that, for every string x of A* of length n, T outputs a string x' of B* in not more than r(n) steps such that x is in L if and only if x' is in M. When problem L is polynomially reducible to problem M we naturally regard M as being at least as difficult as L. There are problems M in NP that are maximally difficult in the sense that every problem in NP is polynomially reducible to M. These problems are called

NP-*complete* and are currently thought by the majority of computer scientists to be intractable.

Cook proved in 1971 the NP-completeness of SAT. His proof amounts to constructing a polynomial reduction of any NP-problem L to the SAT problem. This is reminiscent of Turing's reduction of the halting problem to the first-order tautology problem. Here are Cook's own words:

> Suppose a nondeterministic Turing machine M accepts a set S of strings within time Q(n), where Q(n) is a polynomial. Given an input w for M, we will construct a propositional formula A(w) in conjunctive normal form such that A(w) is satisfiable iff M accepts w. Thus A(w) is easily put in disjunctive normal form (using De Morgan's laws), and A(w) is a tautology if and only if w∉S. Since the whole construction can be carried out in time bounded by a polynomial in |w| (the length of w), the theorem will be proved.

Many important problems become easy for us only after some clairvoyant has guessed a certificate. This is the case of SAT, whose certificates are truth value assignments; KNAPSACK, where the clairvoyant guesses an appropriate selection of stamps; TRAVELING SALESPERSON, where the clairvoyant guesses a travel plan; INTEGER PROGRAMMING, where an integer solution to a system of linear equations with integer coefficients must be guessed... Lacking clairvoyants, these problems are usually solved only for relatively small (if not just toy) input sizes. Indeed, technological progress affording computer speedups does not appreciably increase the threshold of (un)manageable input sizes.

As for COMPOSITE, while up to 2004 the clairvoyant was supposed to guess a nontrivial divisor, the work of Agrawal, Kayal and Saxena has shown that we can dispense with clairvoyants, because COMPOSITE is in P. Interestingly, the polytime algorithm deciding COMPOSITE provides exotic certificates different from nontrivial divisors. A typical certificate for n to be COMPOSITE is an "easy-to-find, easy-to-check proof C" that n is COMPOSITE---but, dismayingly enough, C does not bother to give us any clue about a possible nontrivial divisor of n. The natural/traditional certificates we were looking for when COMPOSITE was only known to be in NP, are "superseded" by certificates like C, given by the polytime algorithm for COMPOSITE and its refinements.

2.6.2 Relative Intractability

One might hope that the borderline between tractability and intractability would be sharp in the following sense:

(i) If a problem is tractable, then sooner or later a polynomial time algorithm for it will be found.
(ii) If a problem is intractable, then a proof of intractability will be found.

Unfortunately, the following may also happen:

(iii) The problem can be shown to have a polynomial time decision procedure, but, due to the non-constructive character of the proof of tractability, nobody has an idea of how to construct a polynomial algorithm for it.
(iv) Embarrassingly enough, the existence of a polynomial time algorithm for the problem might depend on the solution of an open problem in a mathematical area that (prima facie) is fully extraneous to computation theory.

As an instance of (iii), let F be an arbitrary class of graphs with loops and multi-edges and let F have the following property: whenever G is in F and G' is obtainable from a subgraph of G by a sequence of edge contractions, then G' is also in F. Robertson and Seymour proved in their (1983–2004) that membership in F is decidable in polynomial time. Their proof, however, is irremediably non-constructive and gives no hint whatsoever for the construction of a polynomial time algorithm.

As an instance of (iv), the answer to the problem PRIME raised in Gödel's letter to von Neumann was dependent on a fundamental problem to which many mathematicians devoted their best efforts without solving it. Specifically, in (1976) Miller proved that a polynomial time decision method for PRIME exists, provided an extended form of Riemann's hypothesis is true. As we mentioned, only in 2004 was PRIME shown to be in P.

One can also investigate kinds of resources other than Turing time and different measures of input complexity. What might be highly interesting is to switch from the "worst case" scenario adopted throughout this section and consider instead the "average case" complexity of a problem - and try to explain rigorously why many exponential time algorithms work so well in practice. For instance, in linear programming the simplex method is exponential for infinitely many inputs, but linear for almost all (Smale 1983); whereas the ellipsoid method is polynomial (Khachiyan 1979), but quadratic with large coefficients, and hence works worse than the previous method in the average case. It is an open problem whether some mathematical concept can capture tractability in all its aspects. For an informed judgment, conceptual analysis has to go hand in hand with experience derived from mathematical development and computational practice.

2.7 Final Remarks

Throughout this paper we have discussed results and problems concerning Turing-computability, completeness, incompleteness, (un)decidability, tractability, P and NP. Their mutual relationships can be summarized as follows:

What we do not have: By the Turing-Church Undecidability Theorem for first-order logic there is no Turing machine that, given axioms $A_1,...,A_n$ and a sentence B, decides whether there is a proof of B from these axioms.

What we have: By Gödel's Completeness Theorem we have a Turing machine that, given axioms $A_1,...,A_n$ and a sentence B, decides whether there is a SHORT (say cubic in the total length of $A_1,...,A_n$, B) proof of B from these axioms.

What the P/NP problem asks for: Gödel asked in his letter to von Neumann for a Turing machine that, given axioms $A_1,...,A_n$ and a sentence B, decides QUICKLY (in polytime) whether there is a SHORT proof of B from these axioms.

The formulation of each theorem/problem above requires the mathematical notion of "computation step" invented by Turing in 1936/37. Since then, this notion has been used as *the* currency to measure the cost of algorithmic procedures: no inflation has reduced its value; nobody has found an "intuitively computable" function that is not also Turing computable. Every day, computers perform computations that require a relatively small number of steps: in other words, given input length L(j), the output is produced within a number of steps proportional to the square or cube of L(j); i.e., there is a (usually low degree) polynomial p such that the number of steps to produce the output does not exceed p(L(j)). As a consequence, problems with not excessively large input length can be solved in reasonable time. In contrast, SAT, KNAPSACK, TRAVELING SALESPERSON, and INTEGER PROGRAMMING are all NP-complete problems: They have maximum difficulty as a polytime solution for one of them immediately entails the polytime solution for all problems in NP.

Since 1971 the list of NP-complete problems has been enriched by hundreds of new problems coming from such diverse fields as resource allocation, network design, game theory, logic, algebra, geometry, number theory. And yet, SAT is the progenitor of all NP-complete problems, as it is easier to quickly translate every instance of an NP problem into a formula of Boolean logic, than translating it into an instance of KNAPSACK. The reason for this is direct: short Boolean formulas with just the connectives "and", "or", and "not" can be used to describe Turing machine computations. Indeed, for his proof that SAT is NP-complete, given any problem Q in NP, Cook described a polytime function R for every instance x of Q into a conjunctive normal form formula R(x) in such a way that x belongs to Q if and only if R(x) is satisfiable. The availability of this polytime function R is precisely what gives the SAT (as well as the CNFSAT) problem a universal status, within the class of NP problems. Clearly, there is a deep analogy to the Turing-computable reduction of the halting problem to the tautology problem of first-order logic: it was used by Turing to solve Hilbert's *Entscheidungsproblem* in the negative.

Just as Hilbert's decision problem sparked fundamental contributions by Gödel, Turing, Church, and many others, the P/NP problem has great relevance for contemporary mathematics and its applications. For instance, public key cryptography bets on P≠NP, in the hope that the difficulties of NP-complete problems will remain insurmountable forever. If P were equal to NP, much of cryptography, as we under-

stand it today, would no longer work appropriately. Other people bet on P=NP, however, in the hope of great progress in computer science and mathematics. Indeed, many important problems for industry and mathematics would become amenable to computation if P=NP. More importantly, perhaps, P=NP might entail a revolution on the concept of "proof". As a matter of fact, the problem whether a Boolean formula is unsatisfiable, the complementary problem of a polytime decidable problem, would become polytime decidable. Recalling what happened to the COMPOSITE problem after the discovery that it is in P (described at the end of Sect. 2.6.1), we could exhibit "proofs" of unsatisfiability that are immensely shorter and easier to compute than our current certificates, namely, ordinary proofs. In the long run, these certificates would become more and more competitive, albeit possibly less natural, than our traditional proofs. The entire mathematical language might undergo a radical transformation.

Turing's mathematical work is at the crossroad of a fascinating history of ideas, some old and some new, in mutual interaction: proof, formal system; computation, (un)decidability, universal machine; tractability, polytime computability, NP-completeness, reduction. These ideas have received new impetus from Turing's work, and the P/NP problem may be considered as a sort of *Entscheidungsproblem* of our century.

Acknowledgments The authors are grateful to Rossella Lupacchini and Guglielmo Tamburrini for their valuable comments and suggestions towards clarity and understandability. This paper has appeared in earlier form in Italian, as (2014) "Turing, il Matematico", in *Contributi del Centro Linceo Interdisciplinare* "Beniamino Segre", Accademia Nazionale dei Lincei, Rome, Vol. 129, pp. 85–120. Convegno Per il Centenario di Alan Turing Fondatore Dell'Informatica, 22 November 2012, Scienze e Lettere Editore Commerciale, Rome.

References

Ackermann, W. 1928. Zum Hilbertschen Aufbau der reellen Zahlen. *Mathematische Annalen* 99. English translation in "On Hilbert's Construction of the Real Numbers", in J. van Heijenoort, (ed.) (1967), *From Frege to Gödel: A Sourcebook in Mathematical Logic 1879–1931*. 493–507. Cambridge, MA: Harvard University Press.

Agrawal, M.N., N. Kayal, and N. Saxena. 2004. PRIMES is in P. *Annals of Mathematics* 160: 781–793.

Church, A. 1935. An Unsolvable Problem of Elementary Number Theory; Preliminary Report (Abstract). *Bulletin of the American Mathematical Society* 41: 332–333.

———. 1936. An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics* 58: 345–363. Reprinted in Davis, ed. (1965).

Cobham, A. 1964. The Intrinsic Computational Difficulty of Functions. In *Proceedings of the 1964 International Congress for Logic, Methodology, and Philosophy of Science*, ed. Y. Bar-Hillel. Amsterdam: North-Holland.

Cook, S. 1971. The Complexity of Theorem Proving Procedures. In *Proceedings of the Third Annual Acm Symposium on Theory of Computing*, 151–158. New York: ACM.

Davis, M. (ed.). 1965. *The Undecidable, Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Hewlett/NY: Raven Press.

Davis, M. (ed.). 1994. *Solvability, Provability, Definability: The Collected Works of Emil L. Post*. Boston/Basel/Berlin: Birkhäuser.

Davis, M. and W. Sieg 2015. Conceptual Confluence in 1936: Post and Turing. In *Turing's Revolution-The Impact of his Ideas about Computability*, ed. G. Sommaruga and T. Strahm, Boston, Basel, Berlin, Birkhäuser, 3–27.

Dedekind, R. 1872. *Stetigkeit und irrationale Zahlen*. Braunschweig: Vieweg. English translation "Continuity and Irrational Numbers" in W. Ewald, (ed.) (1996), Vol 2, 765–779.

———. 1888. *Was sind und was sollen die Zahlen*. Braunschweig: Vieweg. English translation in W. Ewald, (ed.) (1996), Vol. 2, 790–833.

Edmonds, J. 1965. Paths, Trees, and Flowers. *Canadian Journal of Mathematics* 17: 449–467.

Ewald, W. (ed.) 1996. *From Kant to Hilbert: A Source Book in the Foundations of Mathematics*. 2 Vols. New York: Oxford University Press.

Gandy, R.O. (1980). Church's Thesis and Principles for Mechanisms. In *The Kleene Symposium*, ed. J. Barwise, H.j. Keisler and K. Kunen, 123–148. Amsterdam: North-Holland.

———. 1988. The Confluence of Ideas in 1936. In *The Universal Turing Machine: A Half-Century Survey*, ed. R. Herken, 55–112. New York: Oxford University Press.

Garey, M.R., and D.S. Johnson. 1979. *Computers and Intractability*. San Francisco: W.H. Freeman.

Gödel, K. 193?. Undecidable diophantine proposition. In S. Feferman et al. (eds.), (1995). Kurt Gödel Collected Works Vol. III: Unpublished Essays and Lectures, 164-175. New York: Oxford University Press.

———. 1931. Über formal unentscheidbare Sätze der *Principia Mathematica* und verwandter Systeme I. *Monatshefte für Mathematik und Physik* 38: 173–198. Reprinted with English translation as "On Formally Undecidable Propositions of *Principia Mathematica* and Related Systems I", in S. Feferman et al (eds.). 1986. *Kurt Gödel Collected Works: Vol. I, Publications 1929–1936*, 145–195. New York: Oxford University Press.

———. 1934. On Undecidable Propositions of Formal Mathematical Systems. In *Kurt Gödel Collected Works, Vol. I: Publications 1926–1936*, ed. S. Feferman et al., 346–372. New York/ Oxford University Press.

———. 1946. Remarks Before the Princeton Bicentennial Conference on Problems in Mathematics. In *Kurt Gödel Collected Works, Vol. II: Publications 1938–1974*, ed. S. Feferman et al., 150–153. New York: Oxford University Press.

———. 1963. Poscriptum to Gödel (1931). In *Kurt Gödel Collected Works Vol. I: Publications 1929–1936*, ed. S. Feferman et al., 195. New York: Oxford University Press.

———. 1964. Postscriptum to the 1934 Princeton Lectures. In *Kurt Gödel Collected Works Vol. I: Publications 1929–1936*, ed. S. Feferman et al., 369–371. New York: Oxford University Press.

———. 2003. *Kurt Gödel Collected Works Volume V: Correspondence H-Z*, ed. S. Feferman et al. New York: Oxford University Press.

Herbrand, J. 1931. Sur la non-contradiction de l'arithmétique. *Crelles Journal für die reine und angewandte Mathematik* 166 (1931): 1–8. English translation "On the Consistency of Arithmetic", in J. van Heijenoort (ed.) (1967/2000), pp. 618–628.

Herken, R. (ed.). (1988). *The Universal Turing Machine (A Half-Century Survey)*. Oxford: Oxford University Press.

Hilbert, D., and Bernays, P. 1939/1970. *Grundlagen der Mathematik. II, Die Grundlehren der Mathematischen Wissenschaften* 50, 2nd ed., Berlin/New York: Springer.

Jacobs, K. 1992. *Invitation to Mathematics*. Princeton: Princeton University Press.

Karp, R. 1972. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, ed. R.E. Miller and J.W. Thatcher. New York: Plenum.

Khachiyan, L.G. 1979. A Polynomial Algorithm in Linear Programming, *Doklady Akademii Nauk. SSSR* 244 (1979): 1093–1096. English translation in *Soviet Mathematics Doklady* 20 (1979), 191–194.

Kolmogorov, A., and Uspensky, V. 1958. On the Definition of Algorithm (Russian). *Uspekhi mat. Nauk*. 13(4): 3–28. English translation in *American Mathematical Society Translations* 29 (1963): 217–245.

Levin, L. 1973. Universal Search Problems (Russian). *Problemy Peredachi Informatsii* 9 (3): 115–116. English translation in Trakhtenbrot, B. A. (trans.) (1984). "A Survey of Russian Approaches to *Perebor* (Brute-Force Searches) Algorithms". *Annals of the History of Computing* 6(4): 384–400.

Miller, G.L. 1976. Riemann's Hypothesis and Tests for Primality. *Journal of Computer and System Sciences* 13 (3): 300–317.

Mundici, D. 1983. Natural Limitations of Decision Procedures for Arithmetic with Bounded Quantifiers. *Archive for Mathematical Logic* 23: 37–54.

———. 2012. *Logic, a Brief Course*. Milan: Springer.

Mundici, D., and W. Sieg. 1995. Paper Machines. *Philosophia Mathematica* 3 (3): 5–30.

Péter, R. 1967. *Recursive Functions,* 3rd ed. New York: Academic. 1st ed. (1951), 2nd ed. (revised) 1957, *Rekursive Funktionen*. Budapest: Akadémiai Kiado.

Post, E.L. 1936. Finite Combinatory Processes, Formulation 1. *Journal of Symbolic Logic* 1: 103–105. Reprinted in M. Davis (ed.) (1965), pp. 288–291 and in M. Davis (ed.) (1994), 103–5.

———. 1943. Formal Reductions of the General Combinatorial Decision Problems. *American Journal of Mathematics* 65(2): 199–215. Reprinted in M. Davis (ed.) (1965), 442–460.

———. 1947. Recursive Unsolvability of a Problem of Thue. *Journal of Symbolic Logic* 12(1): 1–11. Reprinted in M. Davis (ed.) (1965), pp. 292–303 and in M. Davis (ed.) (1994), 503–13.

Rabin, M.O. 1960. *Degree of Difficulty of Computing A Function and Hierarchy of Recursive Sets*. Jerusalem: Hebrew University.

Robertson, N., and P. Seymour. 1983. Graph Minors. I. Excluding a Forest. *Journal of Combinatorial Theory* Series B 35: 39–61.

Robertson, N., and P. Seymour 2004. Graph Minors. XX. Wagner's Conjecture. *Journal of Combinatorial Theory* Series B92: 325–357.

Sieg, W. 1994. Mechanical Procedures and Mathematical Experience. In *Mathematics and Mind*, ed. A. George, 91–117. New York: Oxford University Press.

———. 1997. Step by Recursive Step: Church's Analysis of Effective Calculability. Reprinted in *Church's Thesis After 70 years*, ed. A. Olszewski, J. Wolenski, and R. Janusz, 456–485. Ontos Verlag: Frankfurt.

———. 2009. On Computability. In *Philosophy of Mathematics*, ed. A.D. Irvine, 535–630. Amsterdam/Boston/Oxford: Elsevier.

Sieg, W., and J. Byrnes. 1996. K-Graph Machines: Generalizing Turing's Machines and Arguments. In Lecture Notes in Logic 6, ed. P. Hájek, 98–119. New York: Springer.

Skolem, T. 1923. Begründung der elementaren Arithmetik durch die rekurrierende Denkweise ohne Anwendung scheinbarer Veränderlichen mit unendlichem Ausdehnungsbereich. *Skrifter utgit av Videnskapsselskapet i Kristiania. I, Matematisk-naturvidenskabelig klasse* (6): 1–38. English translation "The Foundations of Elementary Arithmetic Established by Means of the Recursive Mode of Thought, Without the Use of Apparent Variables Ranging Over Infinite Domains", in J. van Heijenoort, (ed.) (2000), 302–333.

Smale, S. 1983. On the Average Number of Steps of the Simplex Method of Linear Programming. *Mathematical Programming* 27 (3): 241–262.

Turing, A. M. 1936/37. On Computable Numbers, with an Application to the *Entscheidungsproblem*. *Proceedings of the London Mathematical Society*, 2(42): 230–265. Erratum. 1937. ibid., 43: 544–546.

Turing, A.M. 1950. The Word Problem in Semi-Groups with Cancellation. *Annals of Mathematics* 52: 491–505.

———. 1954. Solvable and Unsolvable Problems. *Science News* 31: 7–23.

van Heijenoort, J. (ed.) 1967/2000. *From Frege to Gödel: A Sourcebook in Mathematical Logic 1879–1931*. Harvard University Press: Cambridge, MA.

Chapter 3
Turing, Gödel and the "Bright Abyss"

Juliette Kennedy

Abstract We consider the development of computability in the 1930s from what we have called the formalism free point of view. We employ a dual conceptual framework: *confluence* together with *grounding*.

In Gödel's view, the Turing analysis of computability grounded that notion.

We follow that idea forward, through to Gödel's introduction, in his 1946 Princeton Bicentennial Lecture, of the concept of ordinal definability in set theory. In particular we trace the influence of Turing's analysis of computability on the provisional program for definability and, to a lesser extent, provability laid out in that lecture by Gödel.

3.1 Introduction

> They have completely forgotten what is a mathematical creation: a vision that decants little by little over months and years, bringing to light the obvious [evident] thing that no one had seen, taking form in an obvious assertion of which no one had dreamed … and that the first one to come along can then prove in five minutes, using techniques ready to hand [*toutes cuites*]. A. Grothendiek, *Recoltes et Sémailles*, 1986

Mathematics: there is that which is taken on faith and that which is proved; there is that which is buried deep inside the heart of the mathematician[1] and that which is wholly public; there is intuition and fact and the "bright abyss" in between; there is the raw, one might say, and the cooked.

I hold up my hand and I count five fingers. I take it on faith that the mapping from fingers onto numbers is *recursive* in the sense of the mathematician's definition of the informal concept, "human calculability following a fixed routine." I cannot *prove* the mapping is recursive—there is nothing to prove! Of course, mathematicians can prove many theorems about recursiveness, *moving forward*, so to speak,

[1] Hadamard writes of nonverbal thought, images that resist statement in words, sudden insight, flashes of inspiration. See (Burgess 2015).

J. Kennedy (✉)
Department of Mathematics and Statistics, University of Helsinki, Helsinki, Finland
e-mail: juliette.kennedy@helsinki.fi

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_3

once the definition of the concept "recursive" has been isolated. Moving backwards is more difficult and this is as it should be: for how can one possibly hope to prove that a mathematical definition captures an informal concept?

This paper is about that one word, *capture*, or more precisely the relation "x captures y," and the question, if y is taken to be computability, definability or provability, does there exist an *adequate* and unique choice of x?

By which criteria do we shape, commit ourselves to, or otherwise assess standards of adequacy? This is the problem of *faithfulness*; the problem of what is *lost* whenever an intuitively given mathematical concept is made exact, or, beyond that, formalized; the problem, in words, of the *adequacy* of our mathematical definitions. It is always present in mathematics, but it is a philosophical problem rather than a mathematical one, in our view, as there is nothing in this idea of "fit" that can be subject to mathematical proof.[2]

Logicians have developed a number of what one might call coping strategies. For example, some have raised the question of intensional adequacy in connection with the Second Incompleteness Theorem. This is because establishing the theorem may turn on the meaning of the (formal) consistency statement as read by the relevant theory—or so Feferman, Detlefsen, Franks and others have argued.[3] In their view, briefly, one should grant the meta-theoretical claim that a theory T cannot prove its own consistency only when there is a sentence which T both "recognizes" as a consistency statement, and which T cannot prove. The criteria for a theory T's "recognizing" its consistency statement as such are met just in case it can be proved in T that its proof predicate satisfies the three Hilbert-Bernays derivability conditions.[4] A consistency statement is nothing but a statement to the effect that something is not provable, which means that the adequacy claim in question rests on the further claim that the provability predicate of T is recognised by T as adequately representing "genuine" provability.

Another coping strategy involves the use of the word *thesis*—a word which serves to, in effect, flag an adequacy claim, if not to bracket it. The Church-Turing Thesis, for example, in its present formulation, equates the class of intuitively computable number-theoretic functions with the class of functions computable by a Turing Machine.[5] Other conceivable theses in mathematics include *Weierstrass's*

[2] Some would claim to the contrary that Turing proved a theorem equating human effective calculability with Turing calculability. See below. The faithfulness problem is only a problem for those who take the view that a sharply individuated concept is present, intersubjectively, in the first place. In his (2009) Hodges takes a different view: "*Turing's Thesis* is the claim that independent of Turing we have an intuitive notion of an effectively computable function, and Turing's analysis exactly captures this class of functions... This kind of claim is impossible to verify. Work like Turing's has a power of creating intuitions. As soon as we read it, we lose our previous innocence."

[3] This is in contrast to the First Incompleteness Theorem, which has nothing to do with the *meaning* of the undecidable sentence G—all that matters in that case is that G is undecidable. See (Feferman 1960/1961), (Detlefsen 2001), (Franks 2009) and (Pudlak 1996).

[4] As given in their (1934/1968).

[5] This is "Thesis T" in (Gandy 1988).

thesis, as some have called it,[6] namely the assertion that the $\varepsilon - \delta$ definition of continuity correctly and uniquely expresses the informal concept; *Dedekind's thesis*, asserting that Dedekind gave the correct definition of the concept "line without gaps"[7]; or, alternatively, asserting that Dedekind gave the correct definition of a natural number; the *area thesis*, asserting that the Riemann integral correctly captures the idea of the area bounded by a curve. *Hilbert's Thesis*, a central though contested claim in foundations of mathematics, refers to the claim that "the steps of any mathematical argument can be given in a first order language (with identity)."[8]

To many working mathematicians the correctness of theses such as Weier strass's is, in Grothendieck's sense of the word, *obvious*—for what more is there to say about continuity than what is encapsulated in Weierstrass's definition of it? The possibility that meaningful concepts may arise in future practice, which may fall under the intuitive notion, while not fitting, e.g. Weierstrass's definition, is simply not considered. In such a case one speaks of *theorems*, not theses. Turing, for example, is thought by some not to have formulated a thesis but rather to have proved a theorem relating human effective computability to computability by means of a Turing Machine.[9]

In sum, there are "theses everywhere," as Shapiro notes in his (2013); not necessarily provable, but at the same time in most cases no longer subject to doubt.

How does the nonlogician (or nonphilosopher) cope with the problem of adequacy? "Things are robust if they are accessible (detectable, measurable, derivable, definable, producable, or the like) in a variety of independent ways," in the words of William Wimsatt.[10] But there is also *grounding*: the idea that, among the class of conceptually distinct precisifications of the given (intuitive) concept, one stands out as being indubitably adequate—as being the *right* idea. This happened, evidently, with the notion of "finite." There are a number of (extensionally) equivalent definitions,[11] but the notion that a set is finite if it can be put into one to one correspondence with a natural number seems to be what "every" mathematician means by the term "finite"—even though the definition is blatantly circular, on its face.[12]

---

[6] See (Shapiro 2013).

[7] See (Sieg 1997).

[8] (Kripke, 2013), p. 81. See also (Burgess 1992), (Rav 2007) and (Shapiro 2013).

[9] *Turing's Theorem* is defined by Gandy as follows: Any function which is effectively calculable by an abstract human being following a fixed routine is effectively calculable by a Turing machine—or equivalently, effectively calculable in the sense defined by Church—and conversely. (Gandy 1988), p. 83.

[10] See (Wimsatt 1994). In computability theory the term "confluence" seems to be preferred over "robustness." See e.g. (Gandy 1988).

[11] In set theory.

[12] This is because the concept of "natural number" is usually defined in terms of finiteness. D.A. Martin expressed a similar thought in his recent (2012): There are various ways in which we can explain to one another the concept of the sequence of all the natural numbers or, more generally, the concept of an ω-sequence. Often these explanations involve metaphors: counting forever; an endless row of telephone poles (or cellphone towers); etc. If we want to avoid metaphor, we can talk of an unending sequence or of an infinite sequence. If we wish not to pack to pack so much

And it also happened after 1936 in connection with the notion of computability, when the Turing analysis of human effective computability was taken by logicians to have solved the adequacy problem in that case.

The number theorist Michael Harris uses the word "avatar" to express a condition of *no grounding*, or *in*adequacy—and within that, the condition of knowing *that*, but not *why*:

> …I suggested that the goal of mathematics is to convert rigorous proofs to heuristics—not to solve a problem, in other words, but rather to reformulate it in a way that makes the solution obvious… "Obvious" is the property Wittgenstein called *übersichtlich*, surveyable. This is where the avatars come in. In the situations I have in mind, one may well have a rigorous proof, but the obviousness is based on an understanding that only fits a pattern one cannot yet explain or even define rigorously. The available concepts are interpreted as the avatars of the inaccessible concepts we are striving to grasp.
>
> One cannot even formulate a problem, much less attempt to solve it; the items (notions, concepts) in terms of which the problem would be formulated have yet to be invented. How can we talk to one another, or to ourselves, about the mathematics we were born too soon to understand?[13]

In this paper we consider the adequacy issue which emerged in the 1930s in connection with the episode in logic and foundations with which this volume is concerned, namely the decanting, in Grothendieck's terms, of the notion of effective computability in the 1930s in the hands of the Princeton logicians on the one hand (namely Gödel, Church, Kleene and Rosser), and Alan Turing on the other.

In particular, we will set what one might call, roughly, the logical approach— an approach which led (and can lead now) to foundational formalism, but which can also be pursued opportunistically and pragmatically, i.e. in the absence of foundational commitments—alongside the more semantically oriented or, as we called it in [31], formalism free point of view.[14]

The term "formalism freeness" refers to a complex of ideas involving the suppression of syntax and the forefronting of semantics in the pursuit of mathematical logic.[15] If we define a logic to be a combination of a list of symbols, commonly

---

into the word "sequence," then we can say that that an ω-sequence consists of some objects ordered so that there is no last one and so that each of them has only finitely many predecessors. This explanation makes the word "finite" do the main work. We can shift the main work from one word to another, but somewhere we will use a word that we do not explicitly define or define only in terms of other words in the circle. One might worry—and in the past many did worry—that all these concepts are incoherent or at least vague and perhaps non-objective.

[13] See (Harris 2015). According to Harris, Pierre Deligne seems to have been the first to use the term "avatar" in its standard mathematical meaning, i.e. "manifestation," or "alternative version."

[14] Foundational formalism was coined in (Kennedy 2013) in order to refer to the idea prevalent in foundations of mathematics in the early part of the twentieth century, associated mainly with the Hilbert Program, of embedding the mathematical corpus into a logical calculus consisting of a formal language, an exact proof concept, and (later) an exact semantics, such that the proof concept is sound and complete with respect to the associated semantics as well as syntactically complete in the sense that all propositions that can be written in the formalism are also decided.

[15] An example are the Abstract Elementary Classes, classes of models in which the elementary submodel relation is replaced by an abstract relation satisfying certain mathematical properties.

called a signature, or vocabulary; rules for building terms and formulas, a list of axioms and rules of proof, and then, usually, a semantics, then a mathematical construction is formalism free when it is in a precise sense insensitive to the underlying logical mode of definition.[16] For example, Gödel's constructible hierarchy L was originally built over first order logic. But as we now know a large class of distinct logics can be used to define *L*. One might say that the dependence of *L* on first order logic is only an apparent one.[17] On the other side of the coin, there is entanglement. For example, zero-one laws for finite structures are sensitive to signature in the sense that relational structures satisfy the beautiful zero-one law, but once one adds function symbols to the language, they do not.[18]

It is important to note that mathematically, formalism freeness is not an all or nothing affair, but rather comes in degrees. As to philosophical commitments, we take the following slogan to heart: "the actual content of mathematics goes beyond any formalization."[19]

Among logicians in whose work formalism freeness, in one form or another, played a role, one should include Emil Post, who called for the return to meaning and truth in the opening of his (1944) and the downplaying of what he called "postulational thinking." Post's concern to expose, unearth and otherwise make fully visible the line dividing "what can be done in mathematics by purely formal means [and] the more important part, which depends on understanding and meaning,"[20] aligns him ideologically with Poincaré. As it turns out, Post's recommendation to develop recursion theory *mathematically*, by stripping off the formalism with which the theory was, in his view, encumbered, led to the formalism free development of recursion theory just along the lines he advocated;[21] and it also gave rise to the development of Post's own model of computability, Post Systems (see below).[22]

---

[16] We use the term logic in this sense, and the term formalism, or formal system, interchangeably.

[17] See (Kennedy 2013) and (Kennedy et al. forthcoming).

[18] In detail, the probability that a random relational structure on the domain {1, ... , n} satisfies a given first order formula tends to either 0 or 1 as n tends to infinity. But if we allow function symbols as part of the language, even the simple sentence $\exists x(f(x) = x)$ has limit probability $1/e$. See Fagin, [14].

[19] The slogan is Colin McLarty's description of Poincaré's view, which he calls "expansive intuitionism" (1997). In her (2014), p.367 Danielle Macbeth describes Yehuda Rav's similar thought, of the distinction between "…a mathematician's proof and a strictly reductive, logical proof, where the former is 'a conceptual proof of customary mathematical discourse, having irreducible semantic content,' and the latter, which Rav calls a derivation, is a syntactic object of some formal system." (subquote in (Rav 1999). She remarks, "Just as Poincaré had argued, to formalise a mathematical proof is to destroy it… " Gödel would echo the thought in notes to himself in 1944 (unpublished, XI): "Is the difference between living and lifeless perhaps that its laws of action don't take the form of "mechanical" rules, that is, don't allow themselves to be "formalized"? That is then a higher level of complication. Those whose intuitions are opposed to vitalism claim then simply: everything living is dead."

[20] (Gandy 1988), p. 93.

[21] As Kripke observed in his recent (2013).

[22] In (1936) Post gave a similar analysis to that of Turing's, "with only Church [(1936)] to hand." See (Kleene 1981a).

In this paper we will review the impact of Turing's formalism free conception of computability on the ideas of Gödel's 1946 Princeton Bicentennial Lecture, also as those ideas are implemented in our (2013). Gödel viewed Turing's analysis of computability as paradigmatic, both as conceptual analysis and as mathematical model, and the effect on his thinking, both in the lecture and subsequently, was substantial.[23] Mathematically, Gödel's "transfer" of the Turing analysis of computability to the case of provability (see below) led to the first formulation of what has come to be known as Gödel's program for large cardinals. In the case of definability this transfer led to the fruitful concept of ordinal definability in set theory.

Philosophically much of Gödel's work (from the mid-1940s onwards) was aimed at formulating a position from which the unrestricted application of the Law of Excluded Middle to the entire cumulative hierarchy of sets could be justified. The project was intertwined with Gödel's Platonism, but it was a goal Gödel shared also with Hilbert, if not necessarily with the so-called Hilbert program (in proof theory). Gödel's *appropriation* of the Turing analysis lent power and plausibility to his search for a logically autonomous perspective,[24] allowing an overview of logical frameworks, while not being entangled in any particular one of them—*for that is what absolute decidability entails*. It is a view from afar; a use of logical formalisms that is localized and opportunistic (rather than global and foundational), in which logical formalisms are used for the purpose of *control*.[25]

As with the notion of "finite set," one cannot think about computability, in its historical context or otherwise, without coming up against the intriguing conceptual anomalies, circularities and the like, which plagued the early attempts to ground the notion of effectivity, and plague them still: on the one hand, computation can be seen as a form of deduction; while on the other hand deduction is easily seen as a form of (non-deterministic) computation. There is also the specter of deviant encodings, which means that there is no principled way to recognize—in an absolute sense—the legitimacy of one's computational model.[26] And while we will focus on Gödel's place in these developments, the difficult question how to rule out deviant encodings is part of a more general point: semantic concepts cannot be eliminated

[23] For a partial list of Gödel remarks on Turing computability see Gödel **193?* (in Gödel 1995); see also the Gibbs Lecture, **1951*, the 1965 Postscriptum to the (Gödel 1934) Princeton lectures. Gödel remarks to Wang in (Wang 1996), and the Wang-Gödel correspondence (in Gödel 2003b) are also relevant. These are discussed below.

[24] The writing of this paper and the particular use in it of the concept of autonomy owes much to the notion of autonomy outlined in Curtis Franks's (2009).

[25] As Gödel wrote to himself in 1943, "A board game is something purely formal, but in order to play well, one must grasp the corresponding content [the opposite of combinatorically]. On the other hand the formalization is necessary for control [note: for only it is objectively exact], therefore knowledge is an interplay between form and content." (Gödel unpublished IX, quoted in Floyd and Kanamori (2016). Absolute provability and absolute *decidability* are directly connected. See (van Atten and Kennedy 2009).

[26] Or so some have argued. See, e.g. (Rescorla 2007) and the rebuttal to it by Copeland and Proudfoot in their (2010).

from our meta-mathematical discourse—even (unexpectedly) in the case of the concept "mechanical procedure."

3.2 Different Notions of Computability Emerge in the 1930s

Many detailed accounts have been given of the emergence of computability in the 1930s, so we do not assume the task of recounting that complex history here. We rather refer the reader to Sieg's (1997, 2005, 2006, 2009), to Gandy's (1988), to Soare's (1996), to Davis's (1982) and to Kleene's (1981a, b). What follows is a condensed history of these events from our point of view, in which we follow the emergence, from the logical, of various mathematical models of human effective computability, culminating with Turing's.

In brief: various lines of thought ran in parallel. Gödel gave the exact definition of the class of primitive recursive functions in his landmark (1931) paper;[27] while in the early 1930s Church had developed the λ-calculus together with Kleene, a type-free and indeed, in Gandy's words, logic-free[28] model of effective computability, based on the primitives "function" and "iteration." The phrase "logic-free" is applicable only from the point of view of the later 1936 presentation of it, as Church's original presentation of the λ-calculus in his (1932) embeds those primitives in a *deductive formalism*, in the Hilbert and Bernays terminology.

Church's original presentation of the λ-calculus was found by Kleene and Rosser to be inconsistent in 1934,[29] which led to Church's subsequent logic-free presentation of it, or so Kleene would imply in his history of computability in the period 1931–1933 (1981a):

> When it began to appear that the full system is inconsistent, Church spoke out on the significance of λ-definability, abstracted from any formal system of logic, as a notion of number theory.[30]

Church's Thesis dates to his suggestion in 1934, to identify the λ-definable functions with the effectively computable ones, a suggestion that would be lent substantial (if not as far as complete) plausibility when Church, Kleene and Rosser together proved the equivalence of λ-definability with computability in the sense of the Herbrand-Gödel equational calculus, also in 1935.[31]

---

[27] Gödel used the term "recursive" for what are now called the primitive recursive functions. The primitive recursive functions were known earlier. Ackermann (1928) produced a function which is *general recursive*, in the terminology of Gödel 1934 Princeton lectures, but not primitive recursive. See also (Péter 1935,1937).

[28] (Gandy 1988), section 14.8. As mentioned above, following Gandy we use the term "effectively computable," or just "effective," to mean "intuitively computable."

[29] About which Martin Davis would remark, unimprovably, "Not exactly what one dreams of having one's graduate students do for one" (Gandy 1988, p. 70).

[30] See (Kleene and Rosser 1935), which relies on Church's (1934).

[31] See below. The proof of the equivalence developed in stages. See (Davis 1982) and (Sieg 1997).

Gödel, when told of Church's suggestion to equate human effective calculability with λ-definability in early 1934, found the proposal "thoroughly unsatisfactory."[32] In a letter to Kleene, Church described Gödel's then suggestion to take a logical approach to the problem:

> His [Gödel's] only idea at the time was that it might be possible, in terms of effective calculability as an undefined notion, to state a set of axioms which would embody the generally accepted properties of this notion, and to do something on that basis. Evidently it occurred to him later that Herbrand's definition of recursiveness, which has no regard to effective calculability, could be modified in the direction of effective calculability, and he made this proposal in his lectures. At that time he did specifically raise the question of the connection between recursiveness in this new sense and effective calculability, but said he did not think that the two ideas could be satisfactorily identified "except heuristically."[33]

We will return to Gödel suggestion later. For the present we note that in Church's lecture on what came to be known as "Church's Thesis" to the American Mathematical Society in 1935, Church used the Herbrand-Gödel equational calculus as a model of effective computation, i.e. recursiveness in the "new sense," rather than the λ-calculus. Perhaps he was swayed by Gödel's negative view of the λ-calculus as a model of effective computability.[34]

In fact Church presented two approaches to computability in the AMS lectures and in his subsequent (1936), based on the lectures: Firstly *algorithmic*, based still on what is now known as the untyped λ-calculus, i.e. the evaluation of the value *fm* of a function by the step-by-step application of an algorithm—and secondly *logical*, based on the idea of calculability in a logic:

> And let us call a function F of one positive integer *calculable within* the logic if there exists an expression f in the logic such that $f(\mu) = \nu$ is a theorem when and only when $F(m) = n$ is true, μ and ν being the expressions which stand for the positive integers m and n.[35]

As an aside, understanding computation as a species of mathematical argumentation is very natural. It is the idea behind the Curry-Howard isomorphism, in which computability and provability are in a precise sense identified; and it has also been recently put forward by Kripke:

> My main point is this: computation is a special form of mathematical argument. One is given a set of instructions, and the steps in the computation are supposed to follow—follow deductively— from the instructions as given.[36]

---

[32] Church, letter to Kleene November 29, 1935. Quoted in (Sieg 1997) and in (Davis 1982).

[33] Church, letter to Kleene as quoted in (Sieg 1997).

[34] The phrase "Church's Thesis" was coined by Kleene in his (1943). About the attitude of Church toward the idea of taking the λ-calculus as canonical at this point Davis remarks, "The wording [of Church's published abstract JK] leaves the impression that in the early spring of 1935 Church was not yet certain that λ-definability and Herbrand-Gödel general recursiveness were equivalent. (This despite Church's letter of November 1935 in which he reported that in the spring of 1934 he had offered to Gödel to prove that 'any definition of effective calculability which seemed even partially satisfactory... was included in λ-definability')" (Davis 1982, p. 10). See Church's (1935) abstract and (Church 1936).

[35] (Church 1936), p. 357.

[36] (Kripke 2013), p. 81.

> In particular, the conclusion of the argument follows from the instructions as given and perhaps some well-known and not explicitly stated mathematical premises. I will assume that the computation is a deductive argument from a finite number of instructions, in analogy to Turing's emphasis on our finite capacity. It is in this sense, namely that I am regarding computation as a special form of deduction, that I am saying I am advocating a logical orientation to the problem.[37]

Viewing computability in terms of logical calculi involves first restricting the class of formal systems in which the computable functions are to be represented. From Church's perspective of the 1930s, the condition in question was (essentially) that the theorems of the formal system should be recursively enumerable. Recursive enumerability is guaranteed here by two things, the so-called step-by-step argument: if each step is recursive then *f* will be recursive; and three conditions: (i) each rule must be a recursive operation, (ii) the set of rules and axioms must be recursively enumerable, (iii) the relation between a positive integer and the expression which stands for it must be recursive.[38]

Church remarks that in imposing the restriction on the formal systems in question, he

> …is here indebted to Gödel, who, in his 1934 lectures already referred to, proposed substantially these conditions, but in terms of the more restricted notion of recursiveness [i.e. primitive recursive JK] which he had employed in 1931, and using the condition that the relation of immediate consequence be recursive instead of the present conditions on the rules of procedure.[39]

We will take up Gödel's 1934 lectures below. Church's step-by-step argument effects a reduction of the notion of effectively computable function to that of calculability in a formal system of the special kind, that it satisfies conditions (i)–(iii), and the Herbrand-Gödel equational calculus has been embedded in it. A number-theoretic function is effective, in other words, if its values can be computed in a formalism which is effectively given in this sense. The argument appears to be circular.[40] Hilbert and Bernays sharpen Gödel's original conditions in their 1939 *Grundlagen der Mathematik II* (1939/1970), in which they present, like Church, a logical calculus rather than a system of the type of Gödel [1934]. The essential requirement of Hilbert and Bernays is that the proof predicate of the logic is primitive recursive. This effects a precise gain: one reduces effectivity now to *primitive* recursion.[41]

---

[37] (Kripke 2013), p. 80. Kripke's point in the paper is that such arguments, being valid arguments, can be, via Hilbert's thesis, stated in a first order language. But then the solution of the *Entscheidungsproblem* follows almost trivially from the Completeness Theorem for first order logic.

[38] Conditions (i)–(iii) are Sieg's formulation of Church's conditions. See (Sieg 1997) p. 165. For Gandy's formulation of the step-by-step argument, see his (1988), p. 77.

[39] (Church 1936), footnote 21, pp. 357–358.

[40] See also Sieg's discussion of the "semi-circularity" of the step-by-step argument in his (1997).

[41] The comparison of Church's (1936) with Hilbert and Bernays's (1939/1970) follows that of Sieg's in his account (2006). See also (Gandy 1988).

In fact, if only because of condition (iii) of Church's argument, namely that the relation between a positive integer and the expression which stands for it must be effective, strictly speaking, the presentation in Hilbert-Bernays (1939/1970) does not solve the circularity problem either. *The plain fact is that any analysis of effectivity given in terms of calculability in a logic, which is itself effectively given, will be vulnerable to the charge of circularity (or, more precisely, infinite regress)*. For if effectivity is explained via a logic which is itself given effectively, one must then introduce a new logic, by means of which the effectivity of the first logic is to be analyzed. It is in keeping with the analysis to assume that the new logic must also be given effectively. But then one needs to introduce a third logic in terms of which the new logic is to be analyzed... and so forth.[42]

In his history of the period Gandy notes that a shift in perspective had set in by 1934 (Church (1936) notwithstanding) due, perhaps, to misgivings of this kind. As he writes, "… in 1934 the interest of the group shifted from systems of logic to the λ-calculus and certain mild extensions of it: the $\lambda - \kappa$ and the $\lambda - \delta$ calculi."[43] Indeed the Herbrand-Gödel equational calculus, like the λ-calculus in the 1936 formulation, is also not a system of logic per se. Nor is Kleene's system presented in his (1936), based on the concept of μ-recursion, a logic; and nor is Post's model of computability presented (also) in 1936 (though based on work he had done in the 1920s).[44] All of these are conceptions of computability given, primarily, mathematically—*but there was no reason whatsoever to believe in their adequacy*.

3.2.1 The "Scope Problem": How General Are the Incompleteness Theorems?

Gödel was among the first to suggest the problem of isolating the concept of effective computability.[45] His interest in the question was driven, at least in part, by the need to give a precise definition of the notion of "formal system"—an important piece of unfinished business as far as the Incompleteness Theorems are concerned, in that it was not clear at the time to which formal systems the theorems apply, outside of *Principia Mathematica*. (We will call this the "scope problem" henceforth.)

As Gödel would realize almost immediately upon proving the Incompleteness Theorems, the solution of the scope problem depends on the availability of a precise

[42] As Mundici and Sieg wrote in their (1994), "The analysis of Hilbert and Bernays revealed also clearly the 'stumbling block' all these analyses encountered: they tried to characterize the elementary nature of steps in calculations, but could not do so without recurring to recursiveness (Church), primitive recursiveness (Hilbert and Bernays), or to very specific rules (Gödel)."

[43] (Gandy 1988), p. 71.

[44] See (Post 1936). For a penetrating analysis of Post's work of the 1920s see (de Mol 2006). Post was apparently aware of the work of the Princeton group, but he was unaware of Turing's. See (Gandy 1988).

[45] See (Gandy 1988), p. 72.

and adequate notion of effective computability. This is because the formal systems at issue in the Incompleteness Theorems, are to be given *effectively*. As Shapiro put the point in his (1990):

> It is natural to conjecture that Gödel's methods [in the Incompleteness Theorems JK] can be applied to any deductive system acceptable for the Hilbert program. If it is assumed that any legitimate deductive system must be effective (i.e., its axioms and rules of inference must be computable), the conjecture would follow from a thesis that no effective deductive system is complete, provided only that it is ω-consistent and sufficient for arithmetic. But this is a statement about all computable functions, and requires a general notion of computability to be resolved.

Indeed, Gödel was careful not to claim complete generality for the Second Incompleteness Theorem in his (1931) paper:

> For this [formalist JK] viewpoint presupposes only the existence of a consistency proof in which nothing but finitary means of proof is used, and it is conceivable that there exist finitary proofs that cannot be expressed in the formalism of *P* (or of *M* and *A*).[46]

Gödel's correspondence with von Neumann in the months following Gödel's verbal reference to the First Incompleteness Theorem in Königsberg in September of 1930, an occasion at which von Neuman was present, reveals a sharp disagreement between the two on the matter. Essentially, von Neumann saw no problem with formalizing "intuitionism," whereas Gödel expressed doubts. As von Neumann wrote to Gödel the following November, "I believe that every intuitionistic [i.e. finitistic JK] consideration can be formally copied, because the "arbitrarily nested" recursions of Bernays-Hilbert are equivalent to ordinary transfinite recursions up to appropriate ordinals of the second number class."[47] And in January 1931 von Neumann responded to the above-cited disclaimer of Gödel's, having seen the galleys of Gödel's (1931) paper:

> I absolutely disagree with your view on the formalizability of intuitionism. Certainly, for every formal system there is, as you proved, another formal one that is … stronger. But intuitionism is not affected by that at all.[48]

Gödel responded to von Neumann, "viewing as questionable the claim that the totality of all intuitionistically correct proofs are contained in *one* formal system."[49] And as he wrote to Herbrand in 1931:

> Clearly, I do not claim either that it is certain that some finitist proofs are not formalizable in *Principia Mathematica*, even though intuitively I tend toward this assumption. In any case, a finitist proof not formalizable in *Principia Mathematica* would have to be quite

[46] (Gödel 1986), p. 195. P is a variant of *Principia Mathematica*.

[47] (Gödel 2003b), p. 339.

[48] (Gödel 2003b), p. 341.

[49] Gödel responded to von Neumann in writing, but his letters seem to have been lost. See (Sieg 2009), p. 548. We know of his response through the minutes of a meeting of the Schlick Circle that took place on 15 January 1931, which are found in the Carnap Archives of the University of Pittsburgh. See Sieg's introduction to the von Neumann-Gödel correspondence in (Gödel 2003b), p. 331.

> extraordinarily complicated, and on this purely practical ground there is very little prospect of finding one; but that, in my opinion, does not alter anything about the possibility in principle.[50]

By the time of Gödel's (*1933*o*) Cambridge lecture Gödel seems to have reversed himself on the question: "So it seems that not even classical arithmetic can be proved to be non-contradictory by the methods of the system A… "[51] Nevertheless Herbrand and Gödel had agreed in their correspondence that the concept of finite computation was itself "undefinable," a view Gödel held through 1934 (and beyond), when he wrote the oft-quoted footnote 3 to the lecture notes of his Princeton lectures:

> The converse seems to be true if, besides recursions according to the scheme (2)[52], recursions of other forms (e.g. with respect to two variables simultaneously) are admitted. This cannot be proved, since the notion of finite computation is not defined, but serves as a heuristic principle.[53]

Looking ahead, Gödel would take a radically different view of the matter later. As he wrote in the amendment to footnote 3 in the 1965 reprinting of his 1934 lectures: "This statement is now outdated; see the Postscriptum (1964), pp. 369–71."[54] In the Postscriptum Gödel indicates that the Turing analysis gave a completely adequate precisification of the concept of finite computation. The Turing analysis would not settle the general question, of course, of the adequacy of any given formalism for the concept of informal provability *überhaupt*. As we will see below, in his 1946 Princeton Bicentennial Lecture Gödel would take a negative view of the matter, expressing the opinion that no single formalism is adequate for expressing the general notion of proof— so not just the notion of finite proof. In fact the view is already expressed in Gödel's 1933 Cambridge lecture (* 1933*o*): "So we are confronted with a strange situation. We set out to find a formal system for mathematics and instead of that found an infinity of systems…".[55]

Returning to the scope problem, although the Second Incompleteness Theorem aims at demonstrating the impossibility of giving a finitary consistency proof in the systems considered, this does not settle the general problem to which formal systems the Incompleteness Theorems apply. As for the problem of isolating the mathematical concept of effectivity, as Sieg notes of the Herbrand correspondence, "Nowhere in the correspondence does the issue of general computability arise."[56]

---

[50] (Gödel 2003b), p. 23.

[51] (Gödel 1995), p. 51. By "system A" Gödel means to refer to systems consisting of finitistically allowable methods. See Feferman's discussion of this point in his (1995) introduction to *1933o*.

[52] i.e., standard recursion.

[53] The claim, the converse of which is being considered here, is the claim that functions computable by a finite procedure are recursive in the sense given in the lectures (Gödel 1986), p. 348.

[54] Gödel's addenda to the 1934 lectures were published in (Davis 1965/2004), pp. 71–73.

[55] (Gödel 1995), p. 47.

[56] (Sieg 2005), p. 180.

By 1934, compelled to "make the incompleteness results less dependent on particular formalisms,"[57] and somewhat at variance with the axiomatic approach he had suggested to Church earlier, Gödel introduced, in his Princeton lectures, the general recursive, or Herbrand-Gödel recursive functions as they came to be known, defining the notion of "formal system" as consisting of "symbols and *mechanical rules* relating to them."[58] Both inference and axiomhood were to be verified by a finite procedure:

> … for each rule of inference there shall be a finite procedure for determining whether a given formula B is an immediate consequence (by that rule) of given formulas $A_1,...A_n$, and there shall be a finite procedure for determining whether a given formula A is a meaningful formula or an axiom.[59]

The Herbrand-Gödel recursive functions are mathematically rather than logically presented. As Gödel remarked in the opening lines of the lecture in connection with the function class, these are "considerations which for the moment have nothing to do with a formal system."[60]

The calculus admits forms of recursions that go beyond primitive recursion. Roughly speaking, while primitive recursion is based on the successor function, in the Herbrand-Gödel equational calculus one is allowed to substitute other recursive functions in the equations, as long as this defines a unique function. (Since for example $f(n) = f(n + 1)$ does not define a unique function.) It was not clear to Gödel at the time, and prima facie it is not clear now, that the schema captures *all* recursions. As Gödel would later write to Martin Davis, "…I was, at the time of these lectures, not at all convinced that my concept of recursion comprises all possible recursions."[61]

Leaving the adequacy question aside for the moment, the equational calculus effects a reduction of one conceptual domain—the strictly proof-theoretic—to another, the mechanical. Gödel's shift of gaze, as it were, was in keeping with the shift in orientation of the Princeton group at the time. The shift may have been driven by the scope problem—for how else to make the concept of formal system exact? Analyzing the concept of "formal system" in terms of the concept of "formal system" itself, would have been no analysis at all.

Returning to Gödel's (1934) lectures, Gödel gives later in the paper the precise definition of the conditions a formal system must satisfy so that the arguments for the incompleteness theorems apply to it. In addition to conditions involving repre-

---

[57] (Sieg 2009), p. 554.

[58] (Gödel 1986), p 349. Emphasis added. Such a mechanistic view of the concept of formal system was not a complete novelty at the time. Tarski conceived of "deductive theory" for example, as "something to be performed." See (Hodges 2008); as Hodges put it, Tarski's view was "that a deductive theory is a kind of activity."

[59] (Gödel 1986), p. 346.

[60] (Gödel 1986), p. 346. This contrasts with Church's initial presentation of the λ-calculus.

[61] Quoted in (Gödel 1986), p. 341. And as Kleene would later write (1981b), "Turing's computability is intrinsically persuasive but λ-definability is not intrinsically persuasive and general recursiveness scarcely so (its author Gödel being at the time not at all persuaded)".

sentability and Church's condition (iii), the relevant restriction Gödel imposed was that

> Supposing the symbols and formulas to be numbered in a manner similar to that used for the particular system considered above, then the class of axioms and the relation of immediate consequence shall be [primitive JK] recursive.[62]

This attaches a primitive recursive characteristic function to each rule of inference and gives a reduction of the concept of "formal system" to recursivity in the sense of the Herbrand-Gödel equational calculus.[63]

As we observed of the analysis of computability Church gave in his (1936), this lends an air of circularity to the analysis (of the notion of formal system). But equally pressing to Gödel may have been the identification of formal provability with computability in the sense of the Herbrand-Gödel equational calculus. Prima facie, there is no compelling reason to suppose even the extensional equivalence of these notions.

We allow ourselves a brief digression in order to mention Gödel's presentation in these lectures of the theorem expressing the undecidable sentence from his 1931 paper in diophantine form—a theorem which remains under-appreciated, if not completely unknown, to the contemporary mathematician.[64]

The proof of the equivalence between the diophantine and the explicitly self-referential statement of Gödel's (1931) paper turns on the fact that the Δ_0 subformula which occurs in the undecidable Π_1 sentence has a recursive characteristic function. We stress again that here, "recursive" would have been meant in the sense of Herbrand-Gödel equational calculus, a fact which may have weakened claims to generality also in that case.

By 1935, Gödel's reflections on computability in the higher order context began to point towards the possibility of a definitive notion of formal system. Nevertheless, an, in Gödel's terminology now, *absolute* definition of effective computability was still missing at that point. As Gödel wrote to Kreisel in 1965:

> That my [incompleteness] results were valid for all possible formal systems began to be plausible for me (that is since 1935) only because of the Remark printed on p. 83 of 'The Undecidable' [Davis, ed. (1965/2004)]... But I was completely convinced only by Turing's paper.[65]

Gödel's "Remark" is contained in the addendum in proof stage to the 1936 abstract "On the Lengths of Proofs" (1936):

---

[62] (Gödel 1986), p. 361.

[63] See also (Sieg 2009), p. 551.

[64] The theorem has an interesting history: at the Königsberg meeting in 1930, in private discussion after the session at which Gödel announced the First Incompleteness Theorem, von Neumann asked Gödel whether the undecidable statement in question could be put in "number-theoretic form," given the fact of arithmetization. Gödel was initially skeptical but eventually proved the diophantine version of the theorem, to his surprise. See (Wang 1996).

[65] Quoted in (Sieg 2006), in turn quoting from an unpublished manuscript of Odifreddi (1990), p. 65.

> It can, moreover, be shown that a function computable in one of the [higher order JK] systems S_i, or even in a system of transfinite order, is computable already in S_1. Thus the notion 'computable' is *in a certain sense* 'absolute', while almost all metamathematical notions otherwise known (for example, provable, definable, and so on) quite essentially depend on the system adopted.[66]

The above passage seems to be the first hint in Gödel writings of preoccupations that would materialize more fully in his 1946 Princeton lecture. The fact that the passage from lower to higher order quantification does not give any new recursive functions is easily seen once Kleene's Normal Form Theorem is in place. Kleene's theorem exhibits a universal predicate, in which a large numerical parameter, which codes the entire computation, is "guessed." The theorem played a crucial role in the acceptance of Church's Thesis, both on Gödel's part, in combination with Turing's work, and generally. This is because Kleene Normal Form is the principal means of establishing *confluence*, the equivalence of different mathematical notions of computability.[67]

As for Gödel's view of the adequacy of the models of computation which had emerged at the time, his premature departure from Princeton in the spring of 1934 put him at some remove from subsequent developments. In particular he would not have been a direct witness to the further empirical work Church, Kleene and Rosser had done, showing that the known effective functions were λ-definable.[68] Gödel had taken ill and would not be on his feet for some time; he had now also turned to the continuum problem.

3.2.2 Turing's Analysis of Computability

In 1936, unbeknownst to the group in Princeton, Turing gave a self-standing analysis of human effective computability and used it to solve the *Entscheidungsproblem*, as Church had solved it just prior.

Rather than calculability in a logic, Turing analyzed effectivity informally but exactly, via the concept of a Turing Machine—a machine model of computability, consisting of a tape scanned by a reader, together with a set of simple instructions in the form of quadruples.[69] More precisely, the analysis consisted of two elements: a conceptual analysis of human effective computation, together with a mathematical precisification of it consisting of rules given by a set of quadruples, as follows: "erase," "print 1," "move left," and "move right."

---

[66] (Gödel 1936), addendum in (Gödel 1986), p. 399. Emphasis added.

[67] See Davis (1982). Kleene's Normal Form Theorem is stated in (Davis 1982) as follows: Every general recursive function can be expressed in the form $f(\mu y(g(x_1 \ldots x_n, y) = 0))$, where f and g are primitive recursive functions.

[68] This point was made by Dana Scott in private communication with the author.

[69] Or quintuples, in Turing's original presentation.

We alluded to circularity in connection with approaches to computability that are centered on the idea of calculability in a logic.[70] The crucial point here is that the Turing analysis of the notion "humanly effectively computable" does not involve the specification of a logic *at all.*[71,72]

The reaction to Turing's work among the Princeton logicians was immediately positive. As Kleene would write in (1981b), "Turing's computability is intrinsically persuasive" but "λ-definability is not intrinsically persuasive and general recursiveness scarcely so (its author Gödel being at the time not at all persuaded)."

As Gödel would later explain to Hao Wang, Turing's model of human effective calculability is, in some sense, perfect:

> The resulting definition of the concept of mechanical by the sharp concept of "performable by a Turing machine" is both correct and unique ...*Moreover it is absolutely impossible that anybody who understands the question and knows Turing's definition should decide for a different concept*.[73]

> The sharp concept is there all along, only we did not perceive it clearly at first. This is similar to our perception of an animal far away and then nearby. We had not perceived the sharp concept of mechanical procedure sharply before Turing, who brought us to the right perspective.[74]

With regard to the scope problem, Turing's conceptual analysis led to its complete solution in Gödel's view. We cited Gödel's 1965 letter to Kreisel above; other evidence to this in the record is to be found in the 1965 publication of the notes of Gödel's (1934) Princeton lectures, this time with many new footnotes, and including the following Postscriptum (1964):

> In consequence of later advances, in particular of the fact that, due to A. M. Turing's work, a precise and unquestionably adequate definition of the general concept of formal system can now be given, the existence of undecidable arithmetical propositions and the non-demonstrability of the consistency of a system in the same system can now be proved rigorously for every consistent formal system containing a certain amount of finitary number theory.

---

[70] By a *logic*, we have meant a combination of a list of symbols, commonly called a signature, or vocabulary; rules for building terms and formulas, a list of axioms and rules of proof, and then, usually, a semantics.

[71] On the formalism or logic-freeness of the Turing model, of course the concept of a Turing Machine may give rise to a logic; or have a logic embedded within it in some very generalized sense, as when one refers to something like "the logic of American politics." But a logic it is not. In (Kennedy forthcoming we observed that if one defines a formalism, or alternatively a logic, as we have done here then with very little mathematical work *each of the informal notions of computability we have considered so far can be seen as generating formal calculi in this sense*, i.e. as an individual, self-standing formalism with its own syntax and rules of proof and so forth. See also Kanamori, "Aspect-Perception and the History of Mathematics" (forthcoming).

[72] Rescorla has argued that the model gives rise to other circularities, based on the problem of deviant encodings. See (Rescorla 2007).

[73] Remark to Hao Wang, in (Wang 1996), p. 203. Emphasis added.

[74] (Wang 1996), p. 205.

> …Turing's work gives an analysis of the concept of "mechanical procedure" (alias algorithm or computation procedure or "finite combinatorial procedure"). This concept is shown to be equivalent with that of a "Turing machine." A formal system can simply be defined to be any mechanical procedure for producing formulas, called provable formulas. For any formal system in this sense there exists one in the [usual] sense that has the same provable formulas (and likewise vice versa)…[75]

Beyond what he said above, Gödel would not explain his endorsement of Turing's machine model at any length in his writings, but rather, simply, point to it from time to time, as the only real example of a fully worked out conceptual (even phenomenological) analysis in mathematics. It is ironic that the "epistemic work," so to speak, that is, the sharpening of intuition usually brought out by formalization, should be brought out by an informal analysis. *But in fact the analysis had to go this way.* One cannot ground the notion of "formal system" in terms of concepts that themselves require the specification of this or that logic or formal system.

For the logicians of the time, then, and for logicians today, the Turing Machine is not just another in the list of acceptable notions of computability—it is the grounding of all of them. Turing held a mirror up to a specific human act: a computor following a fixed routine. The human profile is carried right through to the end, smuggled into mathematical terrain *nec plus*, *nec minus*, *nec aliter*. It is as seamless a fit of the raw and the cooked as ever there was in mathematics, and there is, now, of course, no problem of adequacy. It was "the focus on the user, the human end"[76] that makes the construction so indubitably *right*. The Turing Machine is computation's beating heart.

Gandy saw Turing's isolation from the logical milieu of Princeton as the key to his discoveries:

> It is almost true to say that Turing succeeded in his analysis because he was not familiar with the work of others… The bare hands, do-it-yourself approach does lead to clumsiness and error. But the way in which he uses concrete objects such as exercise book and printer's ink to illustrate and control the argument is typical of his insight and originality. Let us praise the uncluttered mind.[77]

> All the work described in Sections 14.3–14.9[78] was based on the mathematical and logical (and *not* on the computational) experience of the time. What Turing did, by his analysis of the processes and limitations of calculations of human beings, was to clear away, with a single stroke of his broom, this dependence on contemporary experience, and produce a characterization which—within clearly perceived limits—will stand for all time.[79]

What was the role of confluence? Church's Thesis, as originally suggested by Church in 1934, identified effective calculability with λ-calculability, and then with Herbrand-Gödel calculability, after the equivalence between the two was established. Confluence is established in a weak metatheory by means of the Kleene

[75] (Gödel 1986), p. 369.

[76] (Floyd 2013).

[77] (Gandy 1988), p. 83.

[78] I.e. the work on computability of Church and others prior to Turing's (1936/1937) paper.

[79] (Gandy 1988), p. 101.

Normal Form Theorem, also in the case of proving the equivalence of Turing computability with the remaining notions. Prior to Turing's work, the available confluence was seen as only weakly justifying adequacy. Once one has a grounding example in hand this changes—confluence now plays an epistemologically important evidentiary role.

3.2.3 Gödel's Immediate Reaction to Turing's Work

As an historical question, how soon after learning of Turing's work would Gödel see the Turing analysis in such vigorously positive terms as are characteristic of his later communications with Kreisel and Wang, and his 1965 writings? The text *(*193?)*,[80] dating presumably from the years 1936–1939, is informative on this point. In the text, Gödel gives a perspicuous presentation of the Herbrand-Gödel equational calculus. He also improves the result of the 1934 lectures, that undecidable sentences for the formal theories in question can be given in the form of diophantine equations, by showing that the diophantine equations in question are limited in degree and in the number of variables (while not actually computing the bound). His view of Turing's work at the time:

> When I first published my paper about undecidable propositions the result could not be pronounced in this generality, because for — the notions of mechanical procedure and of formal system no mathematically satisfactory definition had been given at the time. This gap has since been filled by Herbrand, Church and Turing. The essential point is to define what a procedure is. Then the notion of formal system follows easily …

Of the version of the Herbrand-Gödel equational calculus presented here, Gödel goes on to say the following: "That this really is the correct definition of mechanical computability was established beyond any doubt by Turing." That is, using the calculus one can enumerate all "possible admissible postulates," in the terminology of the notes; but since one cannot decide which of the expressions defines a computable function, one cannot diagonalize out of the class, "And for this reason the anti-diagonal sequence will not be computable."[81] Gödel then goes on to say, "But Turing has shown more … " The "more" Turing has shown, is the equivalence of the class of functions computable by Herbrand-Gödel calculus, with the class of functions computable by a Turing Machine. Gödel seems to be saying here that the Herbrand-Gödel equational calculus earns adequacy by virtue of its equivalence with Turing's notion—by inheritance, as it were.

We now turn to the program of Gödel's 1946 Princeton Bicentennial Lecture.

[80] (Gödel 1995), p. 164.

[81] (Gödel 1995), p. 168.

3.2.4 Gödel's 1946 Princeton Bicentennial Lecture

Gödel opens the lecture (1946) with the concept of computation, remarking that the concept can be given an absolute and/or, in his words, "formalism independent" definition:

> Tarski has stressed in his lecture the great importance (and I think justly) of the concept of general recursiveness (or Turing computability). It seems to me that this importance is largely due to the fact that with this concept one has succeeded in giving an absolute definition of an interesting epistemological notion, i.e. one not depending on the formalism chosen.[82]

The second sentence is formulated ambiguously. Gödel seems to be referring to two things: on the one hand there is confluence, namely the fact that all of the known mathematical notions of computability that had surfaced since 1934, i.e. the Gödel-Herbrand-Kleene definition (by 1936), Church's λ-definable functions (by 1936), Gödel-Kleene μ-recursive functions (by 1936), Turing Machines (by 1936 (cf. 1936/7)) and Post systems (by 1943),[83] define the same class of functions. But also a second sense of formalism independence emerges in the paper, having to do with *absoluteness* in the sense defined above, namely the stability of a concept belonging to a theory, relative to suitable extensions (of the theory).[84]

In a footnote appended to this first sentence in 1965,[85] Gödel offers the following clarification:

> To be more precise, a function of integers is computable in *any* formal system containing arithmetic if and only if it is computable in arithmetic, where a function f is called computable in S if there is in S a computable term representing f.[86]

Gödel is referring to the fact that any computable (i.e. partial recursive) function representable by a term in a formalism *extending* arithmetic, is already representable in arithmetic, by the same term. The proof depends on the Kleene Normal Form Theorem, referred to above, and generalizes the remark made in Gödel's Speed-up Theorem abstract we cited at the end of Sect. 2, about the absoluteness of computability relative to higher order extensions.

The 1965 clarification was needed, as Gödel may have conflated two different concepts in the opening lines of the lecture. "Formalism independence" taken in the sense of "confluence" applies to a broad range of conceptually distinct notions of computability, and is established by proving the equivalence of the corresponding

[82] (Gödel 1990), p.150.

[83] See, e.g., (Davis 1958).

[84] As Parsons interprets the paragraph in his introduction to (Gödel 1946), Gödel is referring to "the absence of the sort of relativity to a given language that leads to stratification of the notion such as (in the case of definability in a formalized language) into definability in languages of greater and greater expressive power." The stratification is "driven by diagonal arguments" (Parsons (1990), p. 145).

[85] That is, to the version of the lecture published in Davis (ed.) (1965/2004).

[86] See also Gödel addendum to his (1936) discussed in Sect. 2.1.

encodings of these various notions in a suitably chosen metatheory.[87] Whereas in the sense given (possibly) in the lecture and subsequently in the 1965 footnote, absoluteness is a property of a concept which holds only relative to a particular class of formal systems. Strictly speaking, absoluteness in this sense is a restricted form of confluence, applying "piecewise," so to speak, to particular notions of computability, which are only absolute relative to (a particular class of) their extensions. On the other hand Gödel can also be read as using the word "absolute" in the more informal sense of "formalism independence." Certainly he used the word "absolute" in different ways at different times.

In the continuation of the lecture Gödel contrasts the absoluteness and/or formalism independence of the concept of computability, with the failure of the same in the cases of provability and definability:

> In all other cases treated previously, such as demonstrability or definability, one has been able only to define them relative to a given language, and for each individual language it is clear that the one thus obtained is not the one looked for. For the concept of computability, however, although it is merely a special kind of demonstrability or definability, the situation is different. By a kind of miracle it is not necessary to distinguish orders, and the diagonal procedure does not lead outside the defined notion.
>
> This, I think, should encourage one to expect the same thing to be possible also in other cases (such as demonstrability or definability). It is true that for these other cases there exist certain negative results, such as the incompleteness of every formalism... But close examination shows that these results do not make a definition of the absolute notions concerned impossible under all circumstances, but only exclude certain ways of defining them, or at least, that certain very closely related concepts may be definable in an absolute sense.

Just as in the case of computability, where there is an ambient intuitive (epistemological) notion to be made mathematically precise, here too the intuitive notion to be made mathematically precise is, for definability, "comprehensibility by the mind." For provability, the intuitive concept to be made precise is the notion of intuitive consequence.

3.2.5 Provability

We briefly consider Gödel's suggestions regarding provability, before turning to definability.

> Let us consider, e.g., the concept of demonstrability. It is well known that, in whichever way you make it precise by means of a formalism, the contemplation of this very formalism leads to new axioms which are exactly as evident and justified as those with which you started, and that this process of extension can be extended into the transfinite. *So there cannot exist any formalismwhich would embrace all these steps*; but this does not exclude that all these steps; but this does not exclude that all these steps... could be described and collected in some non-constructive way. In set theory, e.g., the successive extensions can be

[87] A rather weak metatheory is required for this.

> most conveniently be represented by stronger and stronger axioms of infinity. It is certainly impossible to give a combinational and decidable characterization of what an axiom of infinity is; but there might exist, e.g., a characterization of the following sort: An axiom of infinity is a proposition which has a certain (decidable) formal structure and which in addition is true. Such a concept of demonstrability might have the required closure property, i.e., the following could be true: Any proof for a set-theoretic axiom in the next higher system above set theory (i.e., any proof involving the concept of truth which I just used) is replaceable by a proof from such an axiom of infinity. It is not impossible that for such a concept of demonstrability some completeness theorem would hold which would say that every proposition expressible in set theory is decidable from the present [*ZFC*] axioms plus some true assertion about the largeness of the universe of all sets.[88]

In brief, some suitable hierarchy of large cardinal assumptions should replace the hierarchy of formal systems generated by the addition of consistency statements to set theory (i.e., passing from *ZFC* to *ZFC* + *Con*(*ZFC*), and then iterating this). Alternatively one could add a satisfaction predicate for the language of set theory, then consider set theory in the extended language, and iterate this. Large cardinal axioms known nowadays can be stated in a first order language. But the proof concept here refers to informal truth, and is thus necessarily informal.

A partial step toward replacing logical hierarchies by infinitary principles would seem to be expressed by the following result of Woodin: in the presence of large cardinals and assuming *CH*, the Σ^2_1 theory of real numbers, i.e., existential statements about sets of reals, is (set) forcing immune, in the sense that their truth cannot be changed by forcing.[89]

Is the concept of provability behind Woodin's result formalism free? An Ω-proof, a proof concept which occurs in Woodin's work in connection with generic absoluteness, is just a universally Baire set of reals. Replacing the concept of a proof by a universally Baire set of reals has an appearance of formalism freeness, somewhat reminiscent of the idea of replacing a formula by a set invariant under automorphisms in the Abstract Elementary Class context.

3.2.6 Definability

In contrast to the difficulties connected with absolute provability, Gödel says that with respect to definability, he can give "somewhat more definite suggestions." And indeed the mathematical content of the lecture mostly concerns the attempt to find an adequate notion of definability in set theory. The term definability here is taken not in the standard technical sense. Finding an absolute characterization of definability would mean to find "an adequate formulation for comprehensibility by our mind."

[88] Emphasis added.

[89] See (Woodin 2010).

It is just here in the lecture that Gödel introduces ordinal definability. The idea is to take the ordinals as already given and then define sets by means of the language of set theory as usual, but with finitely many ordinals as parameters:

> Here you also have, corresponding to the transfinite hierarchy of formal systems, a transfinite hierarchy of concepts of definability. Again it is not possible to collect together all these languages in one, as long as you have a finitistic concept of language, i.e., as long as you require that a language must have a finite number of primitive terms. But, if you drop this condition, it does become possible … by means of a language which has as many primitive terms as you wish to consider steps in this hierarchy of languages, i.e., as many as there are ordinal numbers. The simplest way of doing it is to take the ordinals themselves as primitive terms. So one is led to the concept of definability in terms of ordinals ... This concept should, I think be investigated.[90]

The advantage of doing things this way is that ordinals bequeath their "lawlikeness" to the sets constructed from them, namely the ordinal definable sets. They are "formed according to a law."

Ordinal definability is itself definable in set theory, in contrast to definability simpliciter.[91] For this reason, if one passes to the "next language," i.e., one obtained by adding a truth predicate for statements about ordinal definable sets, one obtains no new ordinal definable sets. In analogy with the computable functions, then, ordinal definability is "non-diagonalizable" (or, in this sense, absolute) as well.

Is there a sense in which ordinal definability is formalism independent? Gödel's goal in these remarks involves replacing a formalism, or more precisely a hierarchy of them generated by the addition of truth predicates, by an axiom or principle of infinity of a special kind: the characterization of the principle must be decidable, and the principle must be true. The principle which is implicit in the concept of ordinal definability—the Levy Reflection Principle—would satisfy both requirements for Gödel.[92] We noted above that Gödel does not state this principle in the lecture; but this is the principle that is used in modern accounts to show that *HOD* is a definable class. The Levy Reflection Principle is not itself an axiom of infinity per se, in fact it is provable; but if it is slightly strengthened and then reflected to some V_α, it becomes an axiom of infinity.

Another possible candidate is briefly considered in the lecture, namely the constructible sets, denoted *L*. The constructible sets are defined as follows:

$$L_{\alpha+1} = \{X \subseteq L_\alpha : X \text{ is definable over} (L\alpha, \in) \text{with parameters}\}$$
$$L_v =_{\alpha<v} L_\alpha \text{ for limit v}$$
$$L =_\alpha L_\alpha .$$

[90] Gödel is actually referring to the *hereditarily ordinal definable* sets in the lecture, denoted *HOD*. A set belongs to *HOD* if its transitive closure is ordinal definable.

[91] Gödel must have known this, judging from these remarks, though modern proofs of this depend on the Levy Reflection Principle, which was only proved in 1960.

[92] The principle says that for every *n* there are arbitrarily large ordinals α such that $V_\alpha \prec_n V$.

In other words, we have a hierarchy

$$L_0 \subseteq L_1 \subseteq ...L_\alpha ... \subseteq L$$

and the universe of constructible sets is the union of all levels of the hierarchy. Gödel introduced this hierarchy for his proof of the relative consistency of the Continuum Hypothesis. The idea is that every level L^α has as few sets as possible. Thus $L^{\alpha+1}$ has just the sets which are actually first order definable (with parameters) on the previous level, and at limit stages L^ν we do not introduce anything new. Accordingly, in the resulting universe L there are as few reals as possible and the Continuum Hypothesis holds. Gödel was able to fulfill technically this plan, a formidable task at the time.

The nice thing about L is that different definitions lead to the same concept-there is a certain amount of formalism freeness in it. For example, L is the closure of the class of all ordinals under the so-called Gödel-functions, a finite list of very simple operations on sets. Also, L is the smallest transitive model of *ZFC* that contains all the ordinals.

L is nondiagonalizable (or absolute) in the following sense: if we form the constructible hierarchy and then add to the language of set theory a predicate for "x is constructible," we do not obtain any new constructible sets.

Gödel sees L as inadequate for the notion definability he seeks in the lecture, for a number of reasons. There is the possible existence of non-constructible sets, i.e. "not all logical means of definition are admitted in the definition of constructible sets." There is also the objection, covering both L and HOD, that those concepts "*were not absolute in the strictest sense, but only with respect to a certain system of things, namely the sets as conceived in axiomatic set theory*."

We return to the constructible hierarchy below. For further analysis of Gödel's remarks on definability the reader is referred to our (2013).

Talk of grounding in connection with the concept of definability in set theory seems premature even now. That confluence led to grounding so quickly in the case of computability, was, as Gödel said, "a kind of miracle" (1946, p. 1, quoted above). But the time frame for definability (not to mention the other case, provability) in set theory will be very different.[93]

3.2.7 Inner Models from Extended Logics

Is it possible to transfer the Turing analysis of computability, as we have called it, to the cases of definability and provability, as Gödel seems to be suggesting in his 1946 lecture? The ingredients of that analysis are *confluence* and *grounding*. We argued

[93] Some of the material in this section was drawn from our (2013). That confluence led to grounding is only accurate from the point of view of the logicians in Princeton. Turing gave the grounding example outright, without confluence in the background.

in [31] that in the case of Gödel's constructible hierarchy L, some small degree of confluence may be in view. The specific application described in that paper and given in detail in [33] involved substituting fragments of second order logic for first order definability in the definition of L.[94] As it turns out, L is not very sensitive to the underlying logic, but is rather very robust in this respect, and the same is true of *HOD*.[95]

More precisely, one starts by fixing a notion of definability, in this case constructibility, and views it as an operator on logics, in the sense of taking a logic as a parameter in the construction of L. We denote the result of applying this operator to a logic $\mathcal{L}^*$ as $L(\mathcal{L}^*)$, defined as follows:

$$L_{\alpha+1}(\mathcal{L}^*) = \{X \subseteq L_\alpha(\mathcal{L}^*) : X \text{ is } \mathcal{L}^* - \text{definable over } (L_\alpha(\mathcal{L}^*), \in) \text{ with parameters}\}$$
$$L_\nu(\mathcal{L}^*) = \bigcup_{\alpha<\nu} L_\alpha(\mathcal{L}^*)$$
$$L(\mathcal{L}^*) = \bigcup_\alpha L_\alpha(\mathcal{L}^*).$$

If $\mathcal{L}^*$ is taken to be first order logic, one obtains the constructible hierarchy. Myhill and Scott showed in their (1971), that if $\mathcal{L}^*$ is taken to be second order logic, the class obtained is HOD.

The results of (Kennedy et al. (forthcoming)) indicate that some degree of confluence is attached to these two structures; *and a logically autonomous perspective becomes available through the parametric treatment of logics*.[96]

We have given the name "Turing Analysis" to an approach which takes a canonical mathematical structure and searches for *confluence*, and then with confluence in evidence searches for a *grounding*.[97] The method can be implemented not just for definability in the sense of L or *HOD*, as was done in [33], but in other settings as well. Thinking beyond definability toward other canonical concepts, one might also consider this varying the underlying logic also in those other contexts. In fact any logical hierarchy, e.g., Kleene's ramified hierarchy of reals is amenable to this treat-

[94] And to some degree, *HOD*, taking the concepts identified by Gödel in the lecture.

[95] Interesting intermediate models, i.e. lying strictly between L and HOD, can also be obtained in this way. An analysis of some of these models is given in Kennedy, Magidor and Väänänen's [33]. For example, if the Magidor-Malitz quantifier $Q_\alpha^{MM,n} x_1,\ldots,x_n \varphi(x_1,,\ldots,,x_n)$ is defined as follows:

$$\mathcal{M} \vDash Q_\alpha^{MM,n} x_1,\ldots,x_n \phi(x_1,\ldots,x_n)$$
$$\Leftrightarrow \exists X \subseteq \mathcal{M}\ (|X| \geq \aleph_\alpha \wedge \forall a_1,\ldots,a_n \in X : \mathcal{M} \vDash \phi(\forall a_1,\ldots,a_n)),$$

then by a result of Magidor, if $\mathcal{L}^*$ is taken to be first order logic with the Magidor-Malitz quantifiers adjoined to it, then the version of L built with Magidor-Malitz logic and with first order logic are the same, assuming $0^\sharp$ exists. See [33] for details.

[96] Finding absolute notions of provability and definability was an explicit concern of Post. See (Post and Davis (ed.) (1994)).

[97] One preserving the so-called human profile.

ment, conceivably. Suitable notions of confluence and grounding must be formulated on a case-by-case basis.

3.3 Conclusion

Michael Harris (2015) asks: "How can we talk to one another, or to ourselves, about the mathematics we were born too soon to understand?" And we answered: with "coping strategies," that is, one in particular: *confluence* together with *grounding*; or more exactly, confluence with the *expectation* of grounding. We saw the schema at work in the development of computability in the 1930s, a development that underwent a radical shift with the emergence of Turing's work. We focussed on Gödel's absorption of the Turing analysis at the time and over the subsequent decade, keeping an eye on the scope problem and the development, in tandem, of the notion of *formal system*; we also considered a methodological shift in Gödel's early work, i.e. the entrenchment of a *generalized form of the Turing analysis*, directed both toward consequence and "comprehensibility by the mind," in the service of what we called logical autonomy.

Gödel's search for autonomy took a particular form: finding absolute definitions of three fundamental epistemological notions: computability, definability and provability. The project imposed, *and imposes on us*, contradictory demands: the logician must control her logic, while at the same time seeking to free herself of it. In this way, he logician is forced to look in two directions at once—toward mathematics done purely conceptually, in the pleasant and comfortable vagueness of natural language; and then toward logical and linguistic exactness.

What interests us here happened on a different, and perhaps smaller stage: certain slight shifts of attention on Gödel's part, and on the part of other logicians working in the 1930s; tentative attempts at autonomy, that, if one is attuned to them in the right way, gain us new logical ground.

Appendix: Deviant Encodings

In a strange twist of fate the essentially *philosophical* problem of deviant encodings, as it is now called, cannot be avoided. That is, Gödel's condition in his 1934 Princeton lectures, that "the relation between a positive integer and the expression which stands for it must be recursive" will *always* introduce circularity, insofar as the condition is a constituent of an all-embracing, or in Gödel's other terminology, absolute notion of computability.

In brief: As a general rule, Turing Machines require an input, and at the end of the computation they generate an output. Inputs and outputs must be encoded, as, after all, Turing Machines operate on finite strings, not natural numbers.

There are, of course, more or less obvious ways to do this. The obvious choice, "unary" encoding, involves counting the number of consecutive ones in the beginning of the tape and declaring that as the input, and, respectively, counting at the end of the computation the number of consecutive ones in the beginning of the tape, and declaring that as the output. As far as one can see, this approach is completely unproblematic! *But one has to accept the effectivity of unary encoding by faith*.

In fact, there are also other, equally obvious encodings. For example, one could represent the input number as a binary string which would then be written on the tape. This should be as computable as unary encoding, and in fact it is even more effective because one gets logarithmic compression. But again the effectivity of the (binary) encoding must be taken on faith.

The problem of "deviant encodings", originally pointed out by Shapiro in his (1982), is that the encoding of input and output data on the tape should be itself computable, devoid of unintended elements which may give the machine noncomputable power. For example, using Copeland and Proudfoot's deviant "mirabilis" output encoding (2010), one can design a Turing Machine that computes the Halting Problem.

Who would question the effectivity of unary encoding? Nobody we know! The problem is that one would like to have a general method for separating computable from noncomputable encodings, rather than a list of acceptable ones. But such a general criterion cannot be given.

Acknowledgments For discussions about definability as well as many other issues connected to this paper I wish to thank especially Menachem Magidor. For very helpful comments and correspondence I am also grateful to John Baldwin, Patricia Blanchette, Juliet Floyd, Curtis Franks, Wilfried Sieg and Jouko Väänänen. This paper was finished while I was a guest of the mathematics department at Hebrew University during November of 2014. I am very grateful for the support of my visit, which enabled me to benefit from the extremely stimulating atmosphere of the logic group in Jerusalem.

References

Ackermann, W. 1928. Zum Hilbertschen Aufbau der reellen Zahlen. *Mathematische Annalen* 99 (1): 118–133.

Burgess, J.P. 1992. Proofs About Proofs: A Defense of Classical Logic. In *Proof, Logic and Formalization*, 8–23. London: Routledge.

———. 2015. *Rigor and Structure*. Oxford: Oxford University Press.

Church, A. 1932. A Set of Postulates for the Foundation of Logic. I, II. *Annals of Mathematics (2)* 33: 346–366.

———. 1934. The Richard Paradox. *American Mathematical Monthly* 41 (6): 356–361.

———. 1935. An Unsolvable Problem of Elementary Number Theory. *Bulletin of the American Mathematical Society* 41: 332–333.

———. 1936. A Note on the *Entscheidungsproblem*. *The Journal of Symbolic Logic* 1: 40–41. (Correction 1:101–102).

Copeland, B.J., and D. Proudfoot. 2010. Deviant Encodings and Turing's Analysis of Computability. *The Journal of Symbolic Logic* 41 (3): 247–252.

Davis, M. 1958. *Computability and Unsolvability*. New York: McGraw-Hill Book Co..

———. 1965/2004. *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. New York: Raven Press/Dover Publications.

———. 1982. Why Gödel Didn't Have Church's Thesis. *Information and Control* 54 (1–2): 3–24.

De Mol, L. 2006. Closing the Circle: An Analysis of Emil Post's Early Work. *The Bulletin of Symbolic Logic* 12 (2): 267–289.

Detlefsen, M. 2001. What Does Gödel's Second Theorem Say? *Philosophia Mathematica. Philosophy of Mathematics, Its Learning, and Its Application. Series III* 9 (1): 37–71.

Fagin, R. 1976. Probabilities on Finite Models. *The Journal of Symbolic Logic* 41 (1): 50–58.

Feferman, S. 1960. Arithmetization of Metamathematics in a General Setting. *Polska Akademia Nauk. Fundamenta Mathematicae* 49: 35–92.

———. 1995. Introductory Note to Gödel *1933. In *Kurt Gödel Collected Works Volume III: Unpublished Essays and Lectures*, ed. S. Feferman et al., 36–44. New York: Oxford University Press.

Floyd, J. 2013. The Varieties of Rigorous Experience. In *The Oxford Handbook of the History of Analytic Philosophy*, ed. M. Beaney, 1003–1042. Oxford: Oxford Univ. Press.

Floyd, J., and A. Kanamori. 2016. Gödel vis-à-vis Russell: Logic and Set Theory to Philosophy. In *Kurt Gödel: Philosopher-Scientist*, ed. G. Crocco and E.-M. Engelen, 243–326. Aix-en-Provence: Presses Universitaires de Provence.

Franks, C. 2009. *The Autonomy of Mathematical Knowledge*. Cambridge: Cambridge University Press.

Gandy, R.O. 1988. The Confluence of Ideas in 1936. In *The Universal Turing Machine: A Half-Century Survey*, ed. R. Herken, 55–112. New York: Oxford University Press.

Gödel, K. 1931. Über formal unentscheidbare Sätze der *Principia Mathematica* und verwandter Systeme, I. *Monatshefte für Mathematik und Physik* 38: 173–198. Reprinted with English translation in (Gödel 1986), 144–195.

———. *1933o. *The Present Situation in the Foundations of Mathematics*. Lecture to Meeting of the Mathematical Association of America and the American Mathematical Society, Cambridge, MA, 29–30 December 1933, in (Gödel 1995), 45–53.

———. 1934. *On Undecidable Propostions of Formal Mathematical Systems*. Lectures at the Institute for Advanced Study, Feb.-May 1934, notes taken by S.C. Kleene and J. B. Rosser. Published in (Gödel 1986), 346–371, which includes the Postscriptum (Gödel 1964).

———. 1936. *Über die Länge von Beweisen*. Abstract in Ergebnisse eines mathematischen Kolloquiums; published with English translation, including an addendum, as "On the Length of Proofs" in Gödel (1986), 397–398.

———. 1946. Remarks Before the Princeton Bicentennial Conference on Problems in Mathematics. In (Gödel 1990), 150–153.

———. *1951o. *Some Basic Theorems on the Foundations of Mathematics and Their Implications*. Josiah Willard Gibbs Lecture, Brown University, 26 December 1951, in (Gödel 1995), 304–232.

———. 1964. *Postscriptum to the 1934 Princeton Lectures*. In (Gödel 1986), 369–370. Originally Published in Davis (ed.) (1965/2004), 71–73.

———. 1986. *Kurt Gödel Collected Works Vol. I: Publications 1929–1936*. New York: Oxford University Press.

———. 1990. *Kurt Gödel Collected Works Volume II: Publications 1938–1974*. New York: Oxford University Press.

———. 1995. *Kurt Gödel Collected Works Volume III: Unpublished Essays and Lectures*. New York: Oxford University Press.

———. 2003a. *Collected Works. IV: Correspondence A-G*. Oxford: Oxford University Press.

———. 2003b. *Kurt Gödel Collected Works Volume V: Correspondence H-Z*. New York: Oxford University Press.

———. unpublished. *Max Phil Notebooks*. Available in Princeton, New Jersey, Firestone Library and the Library of the Institute for Advanced Study, Princeton University.

Harris, M. 2015. *Mathematics Without Apologies: A Portrait of a Problematic Vocation*. Princeton: Princeton University Press.

Hilbert, D., and P. Bernays. 1934/1968. *Grundlagen der Mathematik. I, Die Grundlehren der Mathematischen Wissenschaften 40*. Berlin, New York: Springer Verlag.

———. 1939/1970. *Grundlagen der Mathematik. II, Die Grundlehren der Mathematischen Wissenschaften 50*. Berlin, New York: Springer Verlag.

Hodges, W. 2008. Tarski's Theory of Definition. In *New Essays on Tarski and Philosophy*, 94–132. Oxford: Oxford University Press.

———. 2009. Set Theory, Model Theory, and Computability Theory. In *The Development of Modern Logic*, 471–498. Oxford: Oxford University Press.

Kanamori, A. forthcoming. Aspect Perception and the History of Mathematics. In *Seeing-as and Novelty*, ed. M. Beaney and D. Shaw. New York: Routledge.

Kennedy, J. 2013. On Formalism Freeness: Implementing Gödel's 1946 Princeton Bicentennial Lecture. *The Bulletin of Symbolic Logic* 19 (3): 351–393.

Kennedy, J. forthcoming. Gödel's 1946 Princeton Bicentennial Lecture: An Appreciation. In J. Kennedy, (ed.), Interpreting Gödel, 109–132. Cambridge: Cambridge University Press.

Kennedy, J., M. Magidor, and J. Väänänen (forthcoming). *Inner Models from Extended Logics*.

Kleene, S.C. 1936. A Note on Recursive Functions. *Bulletin of the American Mathematical Society* 42 (8): 544–546.

———. 1943. Recursive Predicates and Quantifiers. *Transactions of the American Mathematical Society* 53: 41–73.

———. 1981a. Origins of Recursive Function Theory. *Annals of the History of Computing* 3 (1): 52–67.

———. 1981b. The Theory of Recursive Functions, Approaching Its Centennial. *American Mathematical Society. Bulletin. New Series* 5 (1): 43–61.

Kleene, S.C., and J.B. Rosser. 1935. The Inconsistency of Certain Formal Logics. *Annals of Mathematics* (2) 36 (3): 630–636.

Kripke, S. 2013. The Church-Turing "Thesis" as a Special Corollary of Gödel's Completeness Theorem. In *Computability: Gödel, Church, and Beyond*, ed. B.J. Copeland, C.J. Posy, and O. Shagrir. Cambridge, MA: MIT Press.

Macbeth, D. 2014. *Realizing Reason*. Oxford: Oxford University Press.

Martin, D. A. 2012. Completeness or Incompleteness of Basic Mathematical Concepts. In *Exploring the Frontiers of Infinity*, ed. P. Koellner.

McLarty, C. 1997. Poincaré: Mathematics & Logic & Intuition. *Philosophia Mathematica. Philosophy of Mathematics, Its Learning, and Its Applications. Series III* 5 (2): 97–115.

Mundici, D., and W. Sieg. 1994. Computability Theory. *Annals of the History of Computing*.

Myhill, J., and D. Scott. 1971. *Ordinal Definability. Axiomatic Set Theory*. Proceedings of the Symposium of Pure Mathematics, Vol. XIII, Part I, University of California, Los Angeles, (1967)). Providence: American Mathematical Society, 271–278.

Odifreddi, P. (ed.). 1990. *About Logics and Logicians—A Palimpsest of Essays by Georg Kreisel, Vol II: Mathematics*. Manuscript.

Parsons, C. 1990. Introductory Note to Gödel (1946). In (Gödel 1990), 144–150.

Péter, R. 1935. Konstruktion nichtrekursiver Funktionen. *Mathematische Annalen* 111 (1): 42–60.

———. 1937. Über die mehrfache Rekursion. *Mathematische Annalen* 113 (1): 489–527.

Post, E.L. 1936. Finite Combinatory Processes, Formulation 1. *Journal of Symbolic Logic* 1:103–105. Reprinted in Davis (1965/2004), 288–291 and in Post, E. L., and Davis, M. (ed.) (1994), 103–105.

———. 1944. Recursively Enumerable Sets of Positive Integers and Their Decision Problems. *Bulletin of the American Mathematical Society* 50: 284–316. Reprinted in Davis (1965/2004), 304–337 and in Post, E. L., and Davis, M. (ed.) (1994), 461–494.

———. 1947. Recursive Unsolvability of a Problem of Thue. *Journal of Symbolic Logic* 12(1): 1–11. Reprinted in Davis (1965/2004), 292–303 and in Post, E. L., and Davis, M. (eds.) (1994), 503–513.

Post, E.L., and M. Davis (eds.). 1994. *Solvability, Provability, Definability: The Collected Works of Emil L. Post*. Boston/Basel/Berlin: Birkhäuser.

Pudlák, P. 1996. On the Lengths of Proofs of Consistency: A Survey of Results. In *Collegium Logicum*, vol. 2, 65–86. Vienna: Springer.

Rav, Y. 1999. Why Do We Prove Theorems? *Philosophia Mathematica. Philosophy of Mathematics, Its Learning, and Its Application. Series III* 7 (1): 5–41.

———. 2007. A Critique of a Formalist-Mechanist Version of the Justification of Arguments in Mathematicians' Proof Practices. *Philosophia Mathematica. Series III* 15 (3): 291–320.

Rescorla, M. 2007. Church's Thesis and the Conceptual Analysis of Computability. *Notre Dame Journal of Formal Logic* 48 (2): 253–280. (electronic).

Shapiro, S. 1982. Acceptable Notation. *Notre Dame Journal of Formal Logic* 23 (1): 14–20.

———. 1990. Review: Stephen C. Kleene, Origins of Recursive Function Theory; Martin Davis, Why Godel Didn't Have Church's Thesis; Stephen C. Kleene, Reflections on Church's Thesis. *Journal of Symbolic Logic* 55 (1): 348–350.

———. 2013. The Open Texture of Computability. In *Computability—Turing, Gödel, Church, and Beyond*, 153–181. Cambridge, MA: MIT Press.

Sieg, W. 1997. Step by Recursive Step: Church's Analysis of Effective Calculability. *The Bulletin of Symbolic Logic* 3 (2): 154–180.

———. 2005. Only Two Letters: The Correspondence Between Herbrand and Gödel. *The Bulletin of Symbolic Logic* 11 (2): 172–184.

———. 2006. Gödel on Computability. *Philosophia Mathematica* 14: 189–207.

———. 2009. On Computability. In *Philosophy of Mathematics*, ed. A.D. Irvine, 535–630. Amsterdam/Boston/Oxford: Elsevier.

Soare, R.I. 1996. Computability and Recursion. *The Bulletin of Symbolic Logic* 2 (3): 284–321.

Soare, R. 2015. *Why Turing's Thesis Is Not a Thesis*. In eds. Sommaruga and Strahm, 297–310.

Sommaruga, G., and T. Strahm, eds. 2015. *Turing's Revolution: The Impact of His Ideas About Computability*, 297–310. Basel: Birhäuser/Springer.

Turing, A. 1936. On Computable Numbers, with an Application to the *Entscheidungsproblsm*. *Proceedings of the London Mathematical Society* 2 (42): 230–265.

van Atten, M., and J. Kennedy. 2009. Gödel's Modernism: On Set-Theoretic Incompleteness, Revisited. In *Logicism, Intuitionism, and Formalism, What has Become of Them?* ed. Sten Lindström, Erik Palmgren, Krister Segerberg, and H. Stoltenberg. New York: Springer.

Wang, H. 1996. *A Logical Journey*. Cambridge, MA: MIT Press.

Wimsatt, W. 1994. The Ontology of Complex Systems: Levels of Organization, Perspectives, and Causal Thickets. *Canadian Journal of Philosophy* 20: 207–274.

Woodin, W.H. 2010. *The Axiom of Determinacy, Forcing Axioms, and the Nonstationary Ideal*. Berlin: Walter de Gruyter GmbH & Co. KG.

Chapter 4
Justified True Belief: Plato, Gettier, and Turing

Rohit Parikh and Adriana Renero

It is so difficult to find the beginning.
Or, better: it is difficult to begin at the beginning.
And not try to go further back.
On Certainty, Ludwig Wittgenstein

Abstract We examine the relationship between the justified true belief (JTB) account of knowledge and Plato's theory about it as expounded in the *Theaetetus*. Considering Socrates' remarks in the *Theaetetus* brings us to some concerns raised by Turing and to Wittgenstein's famous comment *explanations come to an end somewhere*. We present two simple technical results which bear on the question. Finally, we look at the pragmatic aspects of knowledge attributions. In an Appendix we say a few words about Indian epistemology and Gettier problems.

4.1 Introduction

The *justified true belief* account of knowledge is that knowing something is no more than having a justified belief that it is true, and indeed its being true. There is a common impression that the justified true belief (JTB) definition of knowledge is due to Plato and was undermined by Gettier in his (1963) paper.

Gettier himself says, "Plato seems to be considering some such definition at Theaetetus 201, and perhaps accepting one at Meno 98."

R. Parikh (✉)
Brooklyn College and CUNY Graduate Center, City University of New York, New York, NY, USA
e-mail: RParikh@gc.cuny.edu

A. Renero
The Graduate Center, City University of New York, New York, NY, USA
e-mail: arenero@gradcenter.cuny.edu

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_4

The *Stanford Encyclopedia of Philosophy* article on the "Analysis of Knowledge" says.

> Socrates articulates the need for something like a justification condition in Plato's Theaetetus, when he points out that 'true opinion' is in general insufficient for knowledge. For example, if a lawyer employs sophistry to induce a jury into a belief that happens to be true, this belief is insufficiently well-grounded to constitute knowledge.

Others who have attributed the JTB theory to Plato include Artemov and Nogina (2005).

Kevin Meeker (2004) says, "...in the middle part of the twentieth century it appears that philosophers were remarkably unanimous in asserting that knowledge is simply justified true belief."

However, a cursory look at the *Theaetetus* shows that Socrates at least did *not* endorse the JTB theory.

It is the boy Theaetetus (who was a mere 16 years old at the time) and not Socrates who proposes the JTB account after proposing two others, *knowledge as perception* and *knowledge as true belief*.

> Oh, yes, Socrates, that's just what I once heard a man say; I had forgotten but it is now coming back to me. He said that it is true judgment with an account that is knowledge; true judgment without an account falls outside of knowledge.

Socrates subjects Theatetus' assertion to rigorous analysis and finally undermines this third, JTB account, ending with the words,

> …therefore, knowledge is neither perception, nor true judgment, nor an account added to true judgment.

The JTB account of knowledge, rather than being endorsed by Socrates, is explicitly rejected. But what is of interest to us in this talk is Socrates' objection to JTB which is different from that of Gettier and arguably deeper. Gettier's own undermining of JTB went the following route.

Someone justifiably believes A.

She deduces B from A and indeed A implies B.

And B is true. So the belief in B is both justified and true.

However, unfortunately, A is false so that the belief in B, while justified, can't really be considered knowledge.

4.2 The Nature of Justification

Socrates does not go this route but instead asks what a justification might be like (the Greek term here for justification is *logos*, which translates roughly as "account"; the corresponding Sanskrit term is *pramana*, see Sect. 7 below).

An analogy to justification here is an analysis of the first syllable SO of his own name. SO is composed of the two letters S and O and that spelling out is rather like a justification.

Both an analysis and a justification have structure and Socrates points out that the letters S and O, not having structure, cannot have an analysis.

But is it possible to know the syllable SO without knowing the letters S and O? And if not, then how can we rest a knowledge of SO on a knowledge of S and a knowledge of O?

It would seem that issue of the knowledge of the letter S will bring us back to perception, for we know the letter *when we see it*.

Alas, the perception account of knowledge has already been undermined earlier by Socrates. A pig can perceive the letter S but does not know the letter S. So we are left with empty hands.

Let us now abandon Socrates for the moment and go to another, more modern figure, namely Turing in his analysis of computation (1936/7). Turing actually faces the same problem which Socrates faced but in a slightly different form.

A Turing machine reads symbols on squares on its computation tape, writing other symbols in their place, or moving right or moving left to other squares and symbols. These actions are governed by its program which is a set of quadruples (Davis 1958).

Perhaps the machine, when it reads the symbol A in state q, writes a B there and switches to state q'.

But how does the machine know that it is the symbol A which it is reading?

If the Turing machine is a computer, then this is a design problem for the electrical engineer. But if we put *ourselves* in the role of the machine then we are back to Socrates' problem. How do we *know* the symbol A when we see it? And if we do not know, then how can we proceed with the computation?

For instance, there is indeed a Turing machine which recognizes palindromes and would recognize the string ABCBA as a palindrome while rejecting ABCAB. But its ability to recognize ABCBA is parasitical on its ability to recognize the symbol A, and that ability, while we readily grant it, requires an explanation.

Turing himself does face this problem. He restricts himself to finite alphabets on the grounds that an infinite alphabet would deprive the machine of an ability to recognize the symbols on the tape. Here is what he says (1936/7, p. 249):

> I assume then that the computation is carried out on one-dimensional paper, i.e. on a tape divided into squares. I shall also suppose that the number of symbols which may be printed is finite. If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent.

And later (1936/7 p. 250):

> The behaviour of the computer at any moment is determined by the symbols which he is observing,[1] and his "state of mind" at that moment. We may suppose that there is a bound B to the number of symbols or squares which the computer can observe at one moment. If he wishes to observe more, he must use successive observations. We will also suppose that the number of states of mind which need be taken into account is finite. The reasons for this

[1] Note that Turing is neutral between thinking about the computer as a person or as a machine and his mentalistic terminology is defensible.

> are of the same character as those which restrict the number of symbols. If we admitted an infinity of states of mind, some of them will be "arbitrarily close" and will be confused.

Is the problem which Turing discusses the *same as* the problem discussed by Socrates?

Not necessarily, because Turing assumes that the problem is purely geometrical, but the difficulty of recognizing symbols adequately may arise for non-geometric reasons. But Turing does not address the question *how* the machine recognizes symbols even from a finite alphabet.

The issue is that when we reduce complex problems to simpler ones, we do still have the problem of addressing the simpler problems, perhaps the simplest ones that our analysis will come up with. And now we still have to deal with *these*.

4.3 Wittgenstein

In a slightly different context Wittgenstein does (sort of) face this question in his five red apples example in the *Philosophical Investigations* (2009, §1):

> "But how does he know where and how he is to look up the word 'red' and what he is to do with the word 'five'?" ---Well, I assume that he *acts* as I have described. Explanations come to an end somewhere. "But what is the meaning of the word 'five'?" No such thing was in question here, only how the word 'five' is used.

We are left with what looks like a gap. It almost seems like Wittgenstein's answer is an evasion. But we will see that there isn't really a gap.

Let us proceed by recounting a joke.

Someone calls up on the telephone and a boy of five answers.

Can I speak to your father?

He is not here.

Can I speak to your mother?

She is not here either.

All right, will you write them a note that Socrates called?

OK, how do you spell Socrates?

S.O.C.R.A.T.E.S.

There is a long pause and then the child says,

How do you make an S?

The child's problem is the same problem which both Socrates and Turing faced. An explanation or an account or a justification is of no use to someone who does not know the basic elements.

The basic elements are the individual symbols of the alphabet in this context. In other contexts they might be something else. But in any case, an explanation is made up of parts, made up of further parts.

And when we reach bottom, we reach things for which there is no further decomposition. But these elements are furnished for us because we belong to the same species and have similar training. The 5 year-old boy does belong to our species but lacks the corresponding training.

4.4 Agent Based Justification

So let us replace the notion of "justification for p" to "justification for p for agent A", and allow that different agents will need different justifications even for the same conclusion.

Suppose someone asks why people who were brought illegally to the US as children are eligible for some program. Then we can say, "Because Obama ordered it."

Someone else may need "Because Obama ordered it and he is President" and someone else may even require a lesson in the US constitution.

There is no *one size fits all* justification.

There are justifications which suit the occasion. "How do you know it is an S?" can be answered by, "Because I can see it is an S." But "How do you know Obama is the president?" cannot be answered by "Because I can see he is the president."

The first explanation (knowledge as perception) which Theaetetus gave was not wholly wrong. Knowledge *can* coincide with perception on occasion. But having that perception and interpreting it does not require an 'account' or a 'justification'. It requires membership in a community for which explanations have come to an end *because they are not asked for – normally*.

Or consider another, similar case.[2]

Samuel Johnson having argued for some time with a pertinacious gentleman; his opponent, who had talked in a very puzzling manner, happened to say, "I don't understand you, Sir;" upon which Johnson observed, "Sir, I have found you an argument; but I am not obliged to find you an understanding."

We will now present two simple examples of the way in which justification may differ from person to person. Suppose that the law

$$(\forall x)(Px \wedge Qx \wedge Rx \rightarrow Sx)$$

is known to three agents A, B and C. Moreover there is some person D whom they all know who has properties P, Q and R and thus also S.

However A only knows $P_D \wedge Q_D$, B only knows $Q_D \wedge R_D$ and C only knows $R_D \wedge P_D$.

[2] Thanks to Melvin Fitting for telling us about this example.

Then the justifications of S_D will consist of R_D for agent A, P_D for agent B and Q_D for agent C. They will all learn the same fact, but the justifications will differ.

In the example we just gave, there is a single justification $P_D \wedge Q_D \wedge R_D$ which would work for all three but we can easily imagine cases where this will not work.

Suppose that we have instead the law, for a total number of properties N,

$$(\forall x)(\left[\bigwedge P_i\,(x) \colon i \in N\right] \rightarrow S(x))$$

And agents A_i already know some (proper) subset of properties holding of an x:

$$\left(\bigwedge P_j\,(x) \colon j > i\right).$$

Then agent A_i can learn $S(D)$ by learning the remaining properties hold:

$$\bigwedge P_j\,(D) \colon j \leq i$$

and

$$\bigwedge P_j\,(D) \colon j \leq i$$

will be agent A_i's justification for $S(D)$.

Thus different agents will have different justifications. But if their conscious language is finite and they learn about properties in piecemeal ways, then there will be no universal justification that ever can be used.

We suggest that

(a) an explanation will differ from person to person, depending on what that person already knows (their background).
(b) When we get down to the 'atoms' of explanation, then further explanation *could* be asked for, but typically isn't because membership in a community implies that we do *not* ask further questions.

There are no *complete* explanations because every explanation presupposes something. But nevertheless knowledge and justification still work, and can be systematically represented in a logic.

Finally, while we have questioned Gettier's account of Plato, we have no quarrel with Gettier's own result. Gettier and Socrates offer different challenges to the JTB account and both challenges have their merits.

4.5 Pragmatic Encroachment

Pragmatic encroachment can be motivated by intuitions about cases. Jason Stanley's book *Knowledge and Practical Interests* (2005) argues that it is the best explanation of our intuitions about bank cases like these:

> **Low Stakes** Hannah and her wife Sarah are driving home on a Friday afternoon. They plan to stop at the bank on the way home to deposit their paychecks. It is not important that they do so, as they have no impending bills. But as they drive past the bank, they notice that the lines inside are very long, as they often are on Friday afternoons. Realizing that it wasn't very important that their paychecks are deposited right away, Hannah says, I know the bank will be open tomorrow, since I was there just two weeks ago on Saturday morning. So we can deposit our paychecks tomorrow morning.
>
> **High Stakes** Hannah and her wife Sarah are driving home on a Friday afternoon. They plan to stop at the bank on the way home to deposit their paychecks. Since they have an impending bill coming due, and very little in their account, it is very important that they deposit their paychecks by Saturday. Hannah notes that she was at the bank two weeks before on a Saturday morning, and it was open. But, as Sarah points out, banks do change their hours. Hannah says, I guess you're right. I don't know that the bank will be open tomorrow. (Stanley 2005, 34)

The quandary of the couple can be easily understood if we consider a little decision theory.

Consider that we are thinking of performing a procedure.

If B *then do* W *else do* S

and we think that B is true so we do W.

What are the risks?

In the following, Table 4.1, *B* is "Bank is open tomorrow", W is wait, and S is "stand in line." We assume that the preferences are

1. go on Saturday
2. wait in line and
3. go on Monday (least preferred but not considered a disaster)

If p(B) is more than one half then "wait" dominates "stand in line."

Now let us change the payoffs (Table 4.2).

Now "stand in line dominates" unless p(B) > .95

So while "Bank is open" was "known" before, it isn't known any more.

Table 4.1 Low stakes

	B	**¬B**
W	−5	−15
S	−10	−10

Table 4.2 High stakes

	B	**¬B**
W	−5	−100
S	−10	−10

Note that we are using cardinal utilities and subjective probabilities in our explanation. This maneuver is not really justified since people do not have such numbers in mind when they speak.

However, there is at present no convenient way to use *only* ordinal utilities to make a distinction between two cases where

1. B is worse than A
2. B is *much* worse than A

And this implies, again, that we must distinguish between points of view when we individuate justifications.

Low Attributor-High Subject Stakes Hannah and her wife Sarah are driving home on a Friday afternoon. They plan to stop at the bank on the way home to deposit their paychecks. Since they have an impending bill coming due, and very little in their account, it is very important that they deposit their paychecks by Saturday. Two weeks earlier, on a Saturday, Hannah went to the bank, where Jill saw her. Sarah points out to Hannah that banks do change their hours. Hannah utters, "That's a good point. I guess I don't really know that the bank will be open on Saturday". Coincidentally Jill is thinking of going to the bank on Saturday, just for fun, to see if she meets Hannah there. Nothing is at stake for Jill, and she knows nothing of Hannah's situation. Wondering whether Hannah will be there, Jill utters to a friend, "Well, Hannah was at the bank two weeks ago on a Saturday. So she knows the bank will be open on Saturday".

It seems that Jill is making a mistake but a natural one since she does not know about Hannah's high stakes.

This is an issue involving the Premack-Woodruff *Theory of Mind* (1978). Jill does have a theory of mind about Hannah, but it is the wrong one, as is shown by the fact that she attributes to Hannah a false belief. That very possibility turns, however, on her being in a different situation than Hannah is.

4.6 Conclusion

We have surveyed some of the old and more recent observations about the nature of knowledge, and more specifically, the nature of justification, and argued that the notion is agent relative. We also point out that this issue has been discussed by Socrates, Turing, and Wittgenstein and that the responsibility for the much maligned JTB theory of knowledge cannot fairly be laid at the door of Socrates. He was far more subtle than people seem to have noticed

4.7 Addendum

After we wrote this paper and after it was proof-set, we discovered that we had said nothing about the Indian contributions to epistemology (IE from now on). Work in IE begins with Gautama (not the Buddha) around the third century CE and proceeds through various developments to the seventeenth century. Two major figures in IE are Gautama, the third century author of *Nyaya-sutra* and founder of the *Nyaya* school, and Gangesa, the fourteenth century author of *Tattva-chinta-mani* and the founder of *Navya Nyaya* (new Nyaya). (Gautama's *Nyaya* school is sometimes called *Pracina* or ancient Nyaya to distinguish it from Gangesa's later school).

Both the schools mentioned, *Nyaya* and *Navya Nyaya* are Vedic, but there are other Vedic schools as well as some atheist (or non-Vedic) schools, as well as some Buddhists who showed keen interest in knowledge (*prama*). Some Buddhists are presentists, believing that nothing except the present is real. For a fuller list, please consult (Phillips 2015).

However, two persons who seem most aware of Gettier-like problems are Prasastapada in the sixth century and Sri Harsa in the 11th. (All dates are rough).

A common example used is: *Suppose smoke is seen on a mountain, smoke is associated with fire and so one infers fire on the mountain. Actually, what was seen was steam or dust, mistaken for smoke; but as a matter of fact there is a fire on the mountain.*

4.7.1 Does One Know Then That There Is Fire on the Mountain?

Both Prasastapada and Sri Harsa say No, but oddly enough, Gangesa, who followed them both seems to regard at least some Gettier cases as cases of actual knowledge. Gangesa, who is very much a physicalist, only allows certain kinds of errors and some particular ones do not fall in his taxonomy of errors.

Sources of knowledge (*pramana*) are often classified into a) perception b) inference c) testimony and d) similarity. But some Indian philosophers have rejected all of these except for perception. Testimony is suspect because one does not know that the speaker is honest and knowledgeable. If one infers Gx from Fx and $(\forall x)(Fx \rightarrow Gx)$ then the inference is vulnerable to one's being wrong about $(\forall x)(Fx \rightarrow Gx)$.

And even perception might be veridical only when it is non-conceptual. A pre-lingual child who sees a cow "knows" what she sees but cannot tell us what she saw. But an adult saying, "I saw a cow" is not safe from concepts and the ills they are heir to.

Not being experts in this area we will not say more, but will refer the reader to Saha (2003), Ganeri (2007) and Phillips (2015). All three have very kindly helped us with our investigations, but of course any errors are our own.

Dedication This essay is dedicated to the memory of Hilary Putnam.

Acknowledgments Thanks to Aranzazu San Gines for some of the references. Arthur Collins also made a useful suggestion.

References

Artemov, Sergei, and Elena Nogina. 2005. Introducing Justification into Epistemic Logic. *Journal of Logic and Computation* 15(6): 1059–1073.

Burnyeat, M. 1990. *The Theaetetus of Plato*. Trans. M.J. Levett. Indianapolis: Hackett Publishing.

Davis, M. 1958. *Computability and Unsolvability*. New York: McGraw-Hill Book Co..

Ganeri, Jonardon. 2007. Epistemology in *Pracina* and *Navya Nyaya* (review). *Philosophy East and West* 57 (1): 120–123.

Gettier, E.L. 1963. Is Justified True Belief Knowledge? *Analysis* 23 (6): 121–123.

Ichikawa, J.J., and Steup, M. 2013. The Analysis of Knowledge. In *The Stanford Encyclopedia of Philosophy (Fall 2013 Edition)*, ed. E.N. Zalta, URL = http://plato.stanford.edu/archives/fall2013/entries/knowledge-analysis/.

Meeker, K. 2004. *Justification and the Social Nature of Knowledge*, 156–172. LXIX: Philosophy and Phenomenological Research.

Phillips, Stephen. 2015. Epistemology in Classical Indian Philosophy, *The Stanford Encyclopedia of Philosophy* (Spring 2015 Edition), Edward N. Zalta (ed.).

Plato. 1997. *Plato, Complete Works*. Indianapolis: Hackett Publishing.

Premack, D., and G. Woodruff. 1978. Does the Chimpanzee Have a Theory of Mind? *Behavioral and Brain Sciences* 1 (4): 515–526.

Saha, Sukharanjan. 2003. *Epistemology in Pracina and Navya Nyaya*. Kolkata: Jadavpur University.

Stanley, J. 2005. *Knowledge and Practical Interests*. New York: Oxford University Press.

Turing, A.M. 1936/7. On Computable Numbers, with an Application to the Decision Problem. *Proceedings of the London Mathematical Society* 2(42): 230–265.

Wittgenstein, L. 2009. *Philosophische Untersuchungen = Philosophical Investigations*, trans. G.E.M. Anscombe, P.M.S. Hacker and J. Schulte. Malden: Wiley-Blackwell.

Chapter 5
Turing on "Common Sense": Cambridge Resonances

Juliet Floyd

Abstract Turing was a philosopher of logic and mathematics, as well as a mathematician. His work throughout his life owed much to the Cambridge *milieu* in which he was educated and to which he returned throughout his life. A rich and distinctive tradition discussing how the notion of "common sense" relates to the foundations of logic was being developed during Turing's undergraduate days, most intensively by Wittgenstein, whose exchanges with Russell, Ramsey, Sraffa, Hardy, Littlewood and others formed part of the backdrop which shaped Turing's work. Beginning with a Moral Sciences Club talk in 1933, Turing developed an "anthropological" approach to the foundations of logic, influenced by Wittgenstein, in which "common sense" plays a foundational role. This may be seen not only in "On Computable Numbers" (1936/1937) and Turing's dissertation (1939), but in his exchanges with Wittgenstein in 1939 and in two later papers, "The Reform of Mathematical Phraseology and Notation" (1944/1945) and "Solvable and Unsolvable Problems" (1954).

5.1 Introduction

Turing's philosophical attitude has often been distorted by controversies in recent philosophy of mind—above all, those that have associated his thought with computationalist and behaviorist reductionisms. Of course, Turing pioneered philosophical discussion of computational explanation and modelling in such far-flung fields as cognitive science, artificial intelligence, neurological connectionism, and morphogenesis. He also formulated the famed "Turing Test" in an explicitly philosophical essay (1950). His model of computability is regarded by philosophers of mind and language as a standard model of cognition—although its ability adequately to explain or reduce general semantic and representational elements of intelligence and

J. Floyd (✉)
Department of Philosophy, Boston University, 745 Commonwealth Avenue, Boston, MA 02215, USA
e-mail: jfloyd@bu.edu

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_5

thought, as well as physical and evolutionary processes, is increasingly questioned.[1] In all this fray, appreciation of Turing's own *philosophical* sophistication, especially about uncritical notions of meaning and mentality, has tended to be obscured and underplayed.

I will argue that the standard views here—that Turing was an arch-mechanist and reductive computationalist or functionalist, if not a behaviorist and "strong" proponent of AI—are just wrong. Rather than offering general theories of mind, matter, emergence and/or meaning in the traditional sense of a foundation for knowledge, reference, truth and/or ontology, Turing artfully dodged head-on confrontation with such controversies, shifting their flavor. *Un*decidability and *ir*reducibility were central issues for him: how to think about the *limits* of computational, mathematical and philosophical explanations, as well as the consequent need for what logicians and philosophers, as well as computer scientists, call *types*. These are the delimited, surveyable ordering and organizing of objects, concepts, terms, logical particles, definitions, proofs, procedures and algorithms into surveyable wholes. What he provided for future philosophy and science was a way of thinking, a logical framework in which the cost of the absoluteness and ubiquity of computability, as a notion, is flexibility at the basis and on the margins. His universal machine entails a fluidity between hardware, software and data in the realm of computability (cf. Davis Chap. 6). Philosophically Turing also emphasized, however, what we can *do* by way of articulation in ordinary language (cf. Parikh and Renero Chap. 4). We can only grasp his *philosophical* sophistication if we keep what he called "common sense" firmly in mind.

Turing took "common sense" to be a communicable, meaningful level of human discussion awaiting study of its "phraseology". These notions of "common sense" and "phraseology" he developed in conversations with Wittgenstein, as he explicitly acknowledged (See Sect. 5.7 below). Wittgenstein in turn developed these notions in response to an ongoing Cambridge philosophical tradition in logic. Wittgenstein's later philosophy was partly shaped in response to Turing, as some of Turing's work was shaped by him. Here we focus on the Wittgenstein-to-Turing direction. Our point is that Turing's engagement with "common sense", throughout his life, bespeaks his having embedded himself, quite self-consciously, in Cambridge's extended philosophical circles of debate and pedagogy. We shall argue that it influenced several of his most important papers, and ongoing work throughout his life.

For Turing, as for Wittgenstein, "common sense" and "in-the-street" "phraseology" are something received and in turn continually being altered through human discussion, invention, characterization, technology and discovery. With our limited individual cognitive architecture, we humans have long offloaded tasks to calculative routine. Nevertheless, a residue of what is not offloaded has always remained. This end-user conversation is where the *significance* of results, social processes, and logical standards may be contested, redesigned, and re-interpreted. Turing took this residue of "common sense" to be evolving under the pressure of human culture, intellectual development, integration of technology and biology. But its *logical* role

[1] Cf. Putnam (1988), Duwell Chap. 9, Rescorla Chap. 12.

he saw as an irreducible part of the foundations. Already when he was thirteen, he had written, "I always seem to want to make things from the thing that is commonest in nature and with the least waste in energy".[2] "Common sense" and ordinary "phraseology" became a logico-philosophical realization of this idea.

Turing was primarily a mathematician, but one of a very special and reflective kind, and the range and breadth of his contributions were due in part, as he realized, to their deep resonances with philosophical controversies of his day. He himself attempted to articulate and explore this in academic and popular philosophical settings. By tracing the course of his logical and foundational work with an eye on what he was doing *vis-à-vis* his predecessors and his contemporaries at Cambridge, we shall argue that his distinctive, sophisticated attitude—one whose philosophical importance it is easy to overlook or underestimate—comes into view.

Turing's interests in physics, and the extent and limitations of mechanistic explanations of nature, will thus not be our focus. Instead, we shall focus on his foundational interests as they were directed toward the nature of mathematics and logic. Turing's main philosophical contribution here was to recast and clarify the primordial *notion* of logic—by which we shall mean here nothing less than all the different routines, subroutines, patterns and regularities that humans design and follow in their procedures of describing, informing, proving, inferring, arguing about and coming to terms with one another and nature. Turing shifted attention away from the aim of "grounding" logic, mathematical truth, or meaning, and toward an emphasis on *us*—though not primarily, and certainly not only, *via* a theory of mind.

Logic, the technology of deduction, is entangled with what we can write down and take in in the way of patterns and procedures—but also what we can *say* about and do *with* them. In a computational world, applications *via* all kinds of architecture and modelling in physics and mathematics are of great interest, as are cognitive, biological and neurological implementation of routines (whether general purpose or highly specific and modular in character). But these did not frame the fundamental points of departure, philosophically speaking, for Turing. Instead, for him it was always the user perspective, the *interface* that mattered to logic. Here he focused, not only on issues of artifice and accessibility, but also the *reality* of ordinary "phraseology". This seemingly ephemeral and amorphous point is one that he rigorously pursued into the very foundations of logic and mathematics.

The period leading up to "On Computable Numbers" has been a bit of a mystery to those analyzing Turing's intellectual development, insofar there is little direct evidence for how his activities as an undergraduate (1931–1934) pointed toward this great work. We do know from Hodges that Newman's course on foundations of mathematics, in the spring of Turing's first post-graduate Fellowship year (1935) had an immediate impact, introducing him to mathematical logic and setting him onto the *Entscheidungsproblem*; yet we cannot be sure of exactly which lecture courses he attended during his undergraduate years.[3] Nevertheless, there are some

[2] Hodges (1983/2012), p. 19.

[3] Hodges (1983/2012) pp. 63–64 and (1999) pp. 5–7 focus on the importance of Turing's essay "The Nature of Spirit" (1932), where Turing discusses McTaggart, and his reading of Eddington's

important philosophical features of his Cambridge *milieu* that may be traced, and plausibility arguments may be given about these. Our thesis—a development of some of what Turing's biographer Hodges has suggested—is that philosophical ideas around him at Cambridge influenced Turing significantly during his undergraduate years, and especially his discussions with Wittgenstein, who in turn was responding to a longstanding Cambridge tradition of "common sense".[4]

Gödel, struggling in the 1930s with the notion of "general recursive" alongside Church and Kleene, remained focused on devising axiomatic and equational formulations necessary for a logically and mathematically rigorous analysis of "effective calculability", but here "effective" was a broadly "heuristic" notion, deriving from examples intuitively presented, not something philosophically analyzed.[5] By contrast, the philosophical backdrop at Cambridge, in which the *nature* of logic, meaning and mathematics were under discussion, brought to the fore intuitive, human aspects of logic as evolving, purposive technology. Logic was approached, not first and foremost axiomatically, but practically and in thought experiments. These features of the backdrop mattered centrally to Turing. If inspiration is to be found for the distinctive characteristics of his analysis of computability in his undergraduate years 1931–1934, it is here, in the philosophical foundations of logic.

We shall reconstruct in what follows how Turing's notion of "common sense" works its way into his argumentation in several works: a talk on "Mathematics and Logic" to the Moral Sciences Club, December 1933; "On Computable Numbers" (1936/1937); Turing's exchanges with Wittgenstein in Wittgenstein's 1939 Cambridge lectures (1976/1989, hereafter "LFM"); a later, unpublished paper "The Reform of Mathematical Notation" (1944/1945); and his last paper, "Solvable and Unsolvable Problems" (1954). We first shall move thematically (Sects. 5.2 and 5.3) and then chronologically (Sects. 5.4, 5.5, 5.6, 5.7, 5.8 and 5.9), using the arc of Turing's engagement with his Cambridge philosophical *habitus* as the basic line through.

My overall suggestion is that we take philosophical reflection to have been an essential component of Turing's development and contributions, rather than a spectator-sport or a sideline distraction vaguely lying in the penumbra of his thought.

(1929) before coming to Cambridge. Hodges also argues (1983/2012) that Eddington's autumn 1933 course on the methodology of science also had a serious impact on Turing (p. 87), not least setting him on a course of thinking about the Central Limit Theorem, proved in his Fellowship essay. We shall contest none of this.

[4] In (1983/2012) Hodges weaves general aspects of Wittgenstein's thought into his discussion of Turing's work leading up to "On Computable Numbers". In (1999), however, he stresses the absence of any direct positive evidence for Wittgenstein's impact on Turing, arguing that Wittgenstein's 1939 lectures (1976/1989) (LFM) "shed no light on Turing's view of mind and machine" (pp. 22–24). In what follows we agree, but offer a different, though admittedly circumstantial, reconstruction of Turing's development spelling out Hodges' prior hints. As we argue, the nature of mind was not the driving issue behind Turing's initial work in logic, whereas the nature of logic was.

[5] Cf. Mundici and Sieg Chap. 2, Kennedy Chap. 3 Sect. 5.2 above.

New ways of critically thinking about meaning and logic served in this case as a genuine intellectual resource—one all too easy to underestimate as a force.

5.2 The Human Interface

"Common sense", I am arguing, mattered centrally to Turing's logico-mathematical work throughout his life. This is especially evident in his resolution of the *Entscheidungsproblem* in "On Computable Numbers" (1936/1937), his most important and groundbreaking paper. In this paper what Turing *did* was to inject common sense into the heart of the foundations of logic and mathematics. We shall make this out in a more detailed discussion of the paper below, when we revisit the heart of his *specific* argumentation (Sect. 5.5). Here we address the human interface and the importance of "common sense" to the foundations of logic as these figured in the general logico-philosophical context at the time.

In the Hilbert school, the notion of a "general method" for deciding problems—the key notion at issue in the *Entscheidungsproblem*—was framed in terms of the idea of a "finitistic", step-by-step calculation, "derivation" or proof in a formal system of logic. In attacking the intuitive idea of "effective calculability", logicians of the early 1930s generated a more and more crowded and refined set of different logical systems and approaches, finding themselves awash in ideological, philosophical and heuristic discussions. In describing the situation, Post aptly referred to "the forbidding, diverse and alien formalisms" in which the work of logic had been embedded.[6] Mathematical clarification, the theoretical assimilation of this work, would have to take place in *informal* discussion, where the significance and conceptual articulation of the situation could be settled, even if all the reasoning used therein could be, with labor, coded back into the symbolic setting.

By 1935 a desideratum, given the differing precisifications, was an analysis of the general notion of *formal system* itself. In order to gain this, it would not have been enough to write down another, slightly different formalism: one had instead to place the very idea of what such writing down *is* in a new light, one that did not turn on choice of a particular notation, a set of principles, or a specific "metalogic". What was needed was a philosophical clarification of the *first* step, a novel way of thinking. This is what Turing provided: he made the notion of a "formal system" *plain*. Turing's analysis of computability is, as Hodges remarks, "profoundly ordinary".[7]

The vernacular vividness of Turing's analysis in "On Computable Numbers" is widely appreciated. Gödel offered enthusiastic praise when he wrote that Turing offered "the precise and unquestionably adequate definition of the general concept of formal system", one which finally allowed the 1931 incompleteness theorems to be "proved rigorously for *every* consistent formal system containing a certain

[6] Post (1944), p. 284 is looking backward here, summarizing the developments historically.

[7] Hodges (1983/2012), p. 96.

amount of finitary number theory".[8] Gödel had in mind here the mathematically distinctive features of Turing's contribution, first and foremost that the analysis does not depend upon the formalism chosen. This is crucial, not only for demonstrating the precise scope of Gödel's incompleteness results, but also for showing that the notion of "computable" is *absolute*, i.e., it does not change depending upon the particular formal system or set of axioms used. As Gödel had earlier remarked (1946),

> In all other cases treated previously, such as demonstrability or definability, one has been able to define [the fundamental notions] only relative to the given language, and for each individual language it is clear that the one thus obtained is not the one looked for. For the concept of computability, however, although it is merely a special kind of demonstrability or decidability, the situation is different. By a kind of miracle it is not necessary to distinguish orders, and the diagonal procedure does not lead outside the defined notion.[9]

We shall scrutinize the Wittgensteinian air that directly resonates with the heart of Turing's diagonal argumentation in "On Computable Numbers" below (Sect. 5.5). For now, we need to draw out the general role that the themes of "common sense" and "phraseology"[10]*had* to play at the time, anticipating Turing's use of the latter phrases in two of his later works ((1944/1945), (1954); see Sects. 5.7 and 5.8 below).

Each and every individual Turing Machine may be regarded as a precise mathematical object—in fact, a formal system or system of equations, or a collection of quintuples. But, as Gödel noted, the distinctively wide applicability and robustness of Turing's *analysis* of formal logic comes from its other face: the fact that it bypasses entanglement with this or that particular formal language or this or that choice of logic (within a range of recursively axiomatizable, finitary languages of the usual kind). It reduces our general notion of a formal system to an ordinary, informal *comparison* between the step-by-step routine of a human calculator and a machine—in other words, to something belonging to "common sense". In this way Turing avoided entangling himself with dogmatism or ideology, formalism vs. Platonism, intuitionism vs. classical logic, and so on. He showed that it is no part of our general notion of a human being following a (formal) rule that the person must be a classical, or intuitionistic, or any other particular kind of theorist of logic and/or mathematics.[11] As Gandy wrote of Turing's cutting through the knot of formalisms in 1936, "let us praise the uncluttered mind".[12] Turing's comparison allows us

[8] Note added 1964 to Gödel (1934), in Gödel (1986), p. 369; cf. Davis (1982).

[9] Gödel (1946) in his (1990), pp. 150–153. Compare Sieg (2006), especially pp. 472ff and Kennedy (2013, 2014) and Chap. 3.

[10] The idea of "phraseology" occurs explicitly in *The Blue Book* (1965), p. 69; it appears in Wittgenstein's manuscripts and lectures around differing conceptions of numbers and mathematics, cf. e.g. (1999) MS 121, p. 76 (1939), MS 126, p 131 (1942–1943); MS 127, p. 194 (1943) and (1976), pp. 18,91,98. It also occurs in Russell (1919), pp. 141–175, 192. Floyd and Mühlhölzer (unpublished) discuss this notion at length in the context of an analysis of Wittgenstein's annotations to a 1941 edition Hardy's *Course of Pure Mathematics*.

[11] I reconstruct the argument precisely in Floyd (2012b).

[12] Gandy (1988), p. 85.

to recognize his novel construal of the *Entscheidungsproblem* as a remarkable piece of "applied philosophy"[13], a "paradigm of philosophical analysis".[14]

Turing *unvarnished* logic. How did he do this? His basic move was to utilize what Wittgenstein would have been calling, during Turing's undergraduate days—and in dictations widely circulated among mathematics students at Cambridge—the method of a "language-game" or an "object of comparison".[15] Later, perhaps reacting to "On Computable Numbers", Wittgenstein himself explicitly contrasted the use of an informal, everyday, comparative understanding of the foundations of logic with an axiomatic approach, stressing that the former offers a more realistic conception of foundations of logic, whereas the latter invites needless dogmatism and dispute.[16]

Turing's fundamental step in "On Computable Numbers" rests on his drawing a "comparison" between a human computor[17] and a machine (1936/1937 §1). This method is applied at the first step, in Turing's appeal to an ordinary snapshot of a limited portion of human behavior, a calculation made using pen and paper. He was making mathematics out of a "language-game", a simplified snapshot of a portion of human language use designed to elicit from us insights into the workings of logic. This move is, philosophically speaking, *fundamental* to the power of his analysis.

In order to appreciate its philosophical significance, we must understand—contrary to what many have suggested elsewhere—that Turing's analysis did *not* rest on any general theory of language, logic, or mind, even if he utilizes the notion of "state of mind" explicitly in "On Computable Numbers". Just here the Cambridge resonance comes into play. For Wittgenstein's lectures at Cambridge during Turing's years there had stressed, not only the need to extrude issues about inner, mental processes from the foundations of logic; it also brought to the fore an "anthropological" perspective on logic, one that stressed everyday human interfaces with logic in

[13] Davis (1982), p. 14.

[14] Gandy (1988), p. 86.

[15] The method of "comparisons" is evinced in *The Blue Book* (1965) (BB), pp. 5,70 and especially in *The Brown Book* BB §§13,10 (p. 140), §13 (p. 153), §15 (p. 158), §16 (pp. 162, 164), §22, p. 179).

[16] Wittgenstein's use of "*Vergleichsobjekte*" in this particular context of the foundations of logic first appears in (1999) MS 157b, pp. 33–34, drafted in February 1937, just after he would have likely received "On Computable Numbers" from Turing. It reads:

> Our exact language-games are not preparations for a future regimentation of our everyday language, like first approximations without friction and air resistance. This idea leads to *Un*games (*Unspiele)* (Nicod & Russell [JF: who were pursuing the axiomatic approach to logic].) Much more, the language-games stand as *objects of comparison*, which, through similarities and differences should throw light on the relationships of our language.
>
> For the clarity after which we strive is in any case a *complete* one. But this only means that the philosophical problems should *completely* disappear. (cf. (2009) (PI) §130).

[17] Until the late 1940s "computer" referred to a person, often a woman, who carried out calculations and computations in the setting of an office or research facility. Nowadays "computor" is used to make the human user explicit.

informal language as the place where a common basis or "foundation" would have to appear.[18]

Before we delve into this Cambridge backdrop, it is important to qualify our general remarks. As Mundici and Sieg show (Chap. 2), Turing's analysis is subject to perfectly precise axiomatic presentation within the theory of computability. So in saying that Turing connected his analysis with an *informal* model of what we do—thereby doing something of great philosophical interest—we are not saying that he defeated, delimited or circumscribed in any way the mathematician's autonomy to mathematically clarify mathematics: far from it. Turing—and independently Post, with his "workers"[19]—showed how the finistic, constructivistic elements of the proof theory of the Hilbert program, motivated by an idea of rigor that aimed to make mathematical arguments humanly controllable, *had* to be construed in terms of the model of a human computor with limited powers of taking in symbolic configurations *in order to be made mathematically precise*. It had to be done, this self-reflection, this beginning and ending with a snapshot of what we do in doing mathematics.

Turing's specific Cambridge perspective included in this, however, something else: the *general* importance and interest of how we ordinarily phrase ourselves, i.e., "common sense", not only for mathematics, but also for what he would later call its "merging" with logic (1944/1945, cf. Sect. 5.7 below). The mathematical precisification had to be done with minimal fuss and maximal range of applicability. Unlike Post and Gödel, Turing did not believe that his proof rested upon, or even necessarily applied to, limits of the human mind *per se*.[20] Such issues he dodged entirely. Taking up an "anthropological", rather than an ideological perspective on logic, he was able to bypass or transform the issue of psychological fidelity at the foundations, *de*-psychologizing logic, thereby leaving his interest in the mind and its capacities to later works.

Turing's model connected the *Entscheidungsproblem* both to mechanical procedures and to an analysis of the primordial mathematical notion of an algorithm, *via* a kind of snapshot modeling of human mathematical practice. But it therefore also allowed for a generalization across all of logic itself, every recipe of a sort that we may devise and apply, including the recursive aspects of human symbolism and narrative (cf. Winston Chap. 11). It was Turing's insight into the *human purposiveness* of the notion of computability specifically—and logic more generally—that enabled him to boil the relevant notions down to their most vivid, ordinary, essential and simple elements. This turns, however, on Turing's having *not* offered a general definition of logicality, the nature of thought, or the nature of meaning or mind.

[18] Englemann (2013b) explains the importance of this anthropological element.

[19] Cf. Post (1936).

[20] Compare Post's remarks about "psychological fidelity" (1936), p. 105, and Gödel (1972) in (1990), at p. 306, including the introductory note Webb (1990). The reader may contrast Hodges (1999), p. 22 and see Floyd (2012b) for further argumentation. On Gödel vs. Turing, see Copeland and Shagrir (2013), Sieg (2013a).

This was a bold philosophical innovation in the mid-1930s. The point may be made out by contrasting Turing's approach to philosophical problems with that of Carnap, whose "logical syntax" program also aimed to achieve an appropriate philosophical clarification of formal logic within a largely empiricist framework (one that rejected "self-evidence" of logical principles as an *a priori* feature of logic). Carnap took axioms to constitute the meanings of terms in a language, including the logical terms. Rejecting a crude version of verificationism about meaning, he adopted the "principle of tolerance"—that "in logic there are no morals"—and reduced philosophy to the comparative and pragmatic study of the (formal, axiomatized) logical syntax of the language of science.[21] He soon integrated Tarskian "semantics" and the definability of truth in metalanguages into this ambitious program, thereby making the development of scientifically respectable criteria of logicality and analyticity of truth for the languages and metalanguages of science central to the philosophical enterprise.[22]

Turing cut through the whirl very differently from the way that Carnap did, and concertedly so. He used the notion of a "computable number" as a stand-in for the general notion of an effective, indefinitely applicable, mechanical, but humanly operated rule. The comprehensiveness of this treatment—its lack of "morals", if you will—is rooted in his fundamental analytical device, a snapshot of what a human being *does* in computing: scanning a finite sequence of symbols in a step-by-step manner, taking in sequences of them at a glance, moving to a locally differentiated, shareable state, and then operating step-by-step in accordance with finitely articulated, definite "orders" or shareable commands. Turing analyzed what a step in a formal system *is* by thinking through what it is *for*, i.e., what is *done* with it.

Unlike Carnap, Turing was not invested in any attempt to generally characterize the notion of purely logical, meaning-theoretic, or "analytic" truth in terms of formal systems. As we shall soon see, he explicitly and repeatedly bemoaned the idea of forcing the logician or mathematician into a "cast iron system" in which all language would be paraphrased (see Sect. 5.7 below). As opposed to Carnap's proposal that logic (even when relativized to a particular framework) should be able to plan and determine in advance all the purely conceptual, logico-analytic truths about meanings and structures, Turing focused on what it is that we *do* in mathematics or logic when we succeed in setting up a systematic search or effective calculational routine. He was happy to let contexts vary, as we clip one routine out of another and amalgamate it with other routines. Unlike Carnap, Turing valorized "common sense", rather than bemoaning its unclarities: mathematical notation and ordinary language are the result of long evolution of a cooperative enterprise taking place over centuries, and encode "common sense".

These very different philosophical attitudes toward mathematics and logic mattered. The circles and steps among the notions of proof, argument, and step-by-step

[21] Carnap (1934); cf. Carnap et al. (1934) and Carnap (1935).

[22] Carnap's earlier program of "general axiomatics", broached during Gödel's student days with him in the late 1920s, had also been intended to make the proper analysis of symbolic logic (Carnap 2000).

calculability in a logic, easily regarded as prosaic, are rigorous and mathematically tight, demonstrably as well as intuitively so.[23] For throughout Turing's reasoning in "On Computable Numbers", formal derivations, routines, computable processes, alphabets and functions may be regarded as mathematical or logical objects receiving precise characterization. Nevertheless, at the heart of this mathematical analysis remains the fact that the very idea of dynamic, step-by-step motions, actions, processes and states of a Turing machine are given *to* it by our common sense ways of regarding and using it. In fact, implicit throughout this whole range of thinking is something obvious, yet terribly easy to underestimate: *proofs and algorithms are used by humans*.

It is this that Wittgenstein had in mind when, reminiscing 10 years after his 1937 conversations with Turing about "On Computable Numbers", he wrote:

> Turing's 'Machines': these machines are *humans* who calculate. And one might express what he says also in the form of *games* ((1980b) (RPP I) §1095).[24]

What we want to do is to explore how and why Turing embedded himself in a Cambridge context in which this brief philosophical remark could resonate so well with what he had done.

5.3 Turing's Path to "On Computable Numbers", 1931–1935

A distinguished historian of logic, Grattan-Guinness, recently argued that Newman's course in mathematical logic, which Turing followed during Lent Term 1935—in the first year of his graduate fellowship at King's College (1934–1937)—was the sole significant stimulus to Turing's great work, "On Computable Numbers". On this account, without Newman's tutelage, Turing would never have become interested in the foundations of logic at all. Grattan-Guinness writes:

> A possible alternative source of logic for Newman and Turing were a few Cambridge philosophers; but they were largely concerned with revising logicism (especially Frank Ramsey, who went to see Wittgenstein in Austria in 1923 and 1924, and died in 1930). Further, Wittgenstein, back in Cambridge from 1929, was a monist, and so distinguished between what can be said and what can only be shown. So while these philosophers did indeed engage with Turing [Hodges 1983, 152–154], they would not have directed [Turing] toward recursive functions or decision problems. There, by contrast, the Hilbertians stressed the *hierarchy* of "mathematics" and "metamathematics", upon which the actual Turing seized. In the same style Carnap coined "metalogic" in 1931 from seeing it playing a central role in Gödel's [1931] paper, and Tarski was starting to speak of "metalanguage." ((2013), p. 61).

Grattan-Guinness was right to argue that Newman's course was a crucial stimulus for Turing's quick advances in logic, and in two other papers he has established

[23] See Kripke (2013).

[24] See Floyd (2012b) for detailed discussion of this remark, and compare Sieg (2009) whose more general analysis quotes the remark with approval.

the importance of philosophy in general, and Russell in particular, to Newman's own development.[25] Nevertheless, his claims above misdescribe Turing's relation to his *milieu* in Cambridge, especially during his undergraduate years (1931–1934). They certainly err in characterizing Wittgenstein's views.[26] Most importantly, they miscast the importance of Turing's own philosophical stance toward logic, especially toward Carnap's and Tarski's "metalogic", both before and after he wrote "On Computable Numbers". Grattan-Guinness has missed Turing's *philosophy*, and thereby failed to appreciate the depth of his contributions.

Turing did not merely "seize upon" metalogic in the style of Gödel, Tarski and/or Carnap in 1935. Instead he *analyzed* it, and in a manner resonant with certain themes *critical* of "metalogic" stressed frequently among his teachers at Cambridge, especially Wittgenstein. What is most striking about "On Computable Numbers" is what Turing does *not* do. He does *not* ascend to an infinite hierarchy of metalanguages. He does *not* embroil himself in disputes over the law of excluded middle, infinitary objects, or the nature of logic. He does *not* spin off from Gödel's theorem into speculations about the human mind. He does *not* begin by coding up or revising a particular formal or axiomatic system already in use. Issues of consistency and contradiction, even of negation, are *dodged*. So are philosophical disputes. Instead, Turing simply shows what he shows, and directly, in terms of the *human* interface.

In fact, *nothing else would have done*. Turing's analogy, his perspective on "common sense" is a logical *must*. In his remarkable article it is the simplification and the argumentation that matter, and here is where Turing broke through to a novel way of thinking, one it is difficult to imagine Gödel or anyone else having originated.

How did he do it? And why?

5.3.1 Turing's Way in to Logic

In March of 1933, in his second year as an undergraduate, Turing acquired a copy of Russell's *Introduction to Mathematical Philosophy* (1919)—a book that was also a gateway for Gödel and Kleene on their ways in to logic, philosophy and computability.[27] Here Russell engagingly set out, without fuss about symbolism, his treatment of descriptions, the Peano Axioms for arithmetic, the basic logical constructions

[25] Grattan-Guinness (2012a), (b), (2013); Newman (1955), p. 254 states explicitly that "it was in 1935 that [Turing] first began to work in mathematical logic".

[26] P*ace* Grattan-Guinness, by 1931, when Turing arrived at Cambridge, Wittgenstein was an *anti*-"monist", stressing, against his earlier *Tractatus* view, the importance of plurality and variety in systems of logic and grammar. Wittgenstein was hardly engaged in "revising" logicism in 1931–1935, having never embraced it in the higher-order style of Whitehead and Russell: for him logic consisted of tautologies, but mathematics did not; see Floyd (2001a), Weiss (2017).

[27] Russell's (1919) had stimulated Kurt Gödel to turn toward logic from physics at the age of 19 (in a seminar of Moritz Schlick's on the book in Vienna 1925–1926), and had been read by Stephen Kleene before he attended Church's seminar on logic at Princeton in the fall of 1931–1932. Cf. Floyd and Kanamori (2016); Kleene (1981).

of number, the fundamentals of set theory and the theories of order and types, including an account of the paradoxes. It excited Turing; he discussed it for hours with a friend.[28]

Significantly, the final chapter of the book, "Mathematics and Logic", was purely philosophical. Here Russell alluded to Wittgenstein's (then) novel conception of logic as "tautologous". Logicist that he was, Russell equated "mathematics" to "logic" without argument, an equation Wittgenstein never accepted.[29] Offering no clear explanation or defense of "tautology[30]", Russell wrote:

> It is clear that the definition of "logic" or "mathematics" must be sought by trying to give a new definition of the old notion of "analytic" propositions. Although we can no longer be satisfied to define logical propositions as those that follow from the law of contradiction, we can and must still admit that they are a wholly different class of propositions from those that we come to know empirically. They all have the characteristic which, a moment ago, we agreed to call "tautology".
>
> …For the moment I do not know how to define "tautology". [Note: The importance of "tautology" for a definition of mathematics was pointed out to me by my former pupil Ludwig Wittgenstein, who was working on the problem. I do not know whether he has solved it, or even whether he is alive or dead.] (1919, p. 206)

This dramatic ending, calling for a reworking of the traditional distinction between "analytic" and "synthetic" truth (i.e., truth-in-virtue-of-concepts-or-meaning vs. truth-in-virtue-of-fact), would not have escaped Turing's notice. Nor, fatefully enough, would the notion of "tautology". For by 1935 or 1936, having taken Newman's course, Turing would import the use of a tautology-like construction into the heart of his argumentation in "On Computable Numbers" (Sect. 5.5 below). Turing would thus vindicate Russell's suggestion, drawn from Wittgenstein in 1918, that appeal to the laws of contradiction, excluded middle and bivalence are no longer sufficient as a basis for an analysis of logic, whereas the idea of an empty, senseless, repetitive remark "saying the same thing over again" (*tauto- logoi*) *is*.

But why would Turing have been reading Russell at all? Two possible answers present themselves.

The first may be gleaned from Turing's annotated copy of Littlewood's *Elements of the Theory of Real Functions* (1926). In his preface Littlewood explained that he had "aimed at excluding as far as possible anything that could be called philosophy" from his presentation, hoping to "inculcate the proper attitude of enlightened

[28] Hodges (1983/2012), p. 85; (1999), p. 6.

[29] This identification of mathematical sentences with tautologies occurred as a logicist appropriation of Wittgenstein's thought in the Vienna Circle, but also in Cambridge. On January 24, 1941, G. H. Hardy gave a talk to the Moral Sciences Club on "Mathematical Reality," §§20–22 of his book *A Mathematician's Apology* (1940), a book Wittgenstein would call "miserable", probably because it took so little account of his philosophical criticisms (1999, 124, p. 35, from 1941). Mays (1967, p. 82) recalls: "Hardy mentioned that he did not accept Wittgenstein's view that mathematics consisted of tautologies. Wittgenstein denied that he had ever said this, and pointed to himself saying in an incredulous tone of voice, 'Who, I?'" (cf. Klagge and Nordmann eds. (2003), p. 336.) For discussion of Wittgenstein's resulting annotations to Hardy's *Course of Pure Mathematics* (1941), see Floyd and Mühlhölzer (unpublished).

[30] Dreben and Floyd (1991).

simple-mindedness" in his students "by concentrating attention on matters which are abstract but not complicated".[31] But, conceding some readers might be interested in the foundations of mathematics, he recommended that the reader consult Russell's *Introduction to Mathematical Philosophy*, pointing out that its content would be necessary for understanding his presentation of the Peano axioms and the theory of types. Turing's annotations to the inside cover of his copy of Littlewood use the idea of a "propositional function" as well as logical notation.[32] So it is plausible to suppose that Turing turned toward logic in the spring of 1933, during his second undergraduate year, and that this was his introduction to, or motivation for, his study of the theory of types.

There is something else even more interesting in the preface and first sections Littlewood's lecture notes. Lamenting the tendency of modern mathematics papers to eliminate, in their terseness, any discussion of the "*point*" of the subject matter, Littlewood suggests that a lecture, or lecture notes, are perhaps just the sort of place for such "provisional nonsense". He picks the example of his presentation of the Burali-Forti paradox in terms of types, and states that he thinks it possible to

> …make both the underlying ideas seem intuitive and the official proof natural. The infinitely greater flexibility of speech enables me here to do without a blush what I shrink from doing in print.

Moreover Littlewood emphasizes that with respect to certain undefined or "logical" terms ("proposition" for example),

> …we hold ourselves free to choose that mode of expression in ordinary language which is most familiar, vivid, or convenient (1926, p. 2).

Littlewood thereby explicitly refuses to concede "to the cantankerous [logicist or formalist] reader" (p. 2) explicit definitions at all points, insisting that

> No line can strictly be drawn between mathematics and logic, and we are merely using a more or less popular distinction to indicate our starting point (1926, p. 1).

Here, we may say, is a hint at the direction in which Turing's own sensibility would go. The "point" of a mathematical conceptualization needs to be discussed. If "nonsense" emerges in the course of communication, it might only reflect a "provisional" situation, further articulable in terms of natural, familiar, ordinary language.

There is a second possible answer as to why Turing would have been reading Russell in the spring of 1933. Wittgenstein's course "Philosophy for Mathematicians" was given 1932–1933 (Turing's second undergraduate year), and in the beginning of fall 1933 (in the first term of Turing's third and final undergraduate year). It may have influenced Turing (directly or indirectly), drawing him toward logic and foundations of mathematics. Wittgenstein had studied with Littlewood in 1908–1909 and been given his own Cambridge fellowship (1930–1936) largely on the basis of Littlewood's positive assessment of his philosophical work, made after several dis-

[31] Wittgenstein discusses this Prefatorial Remark in his 162a, pp. 15–18.

[32] Turing Digital Archive, AMT/B/46, http://www.turingarchive.org/browse.php/B/46.

cussions; they shared some students and Wittgenstein invited them to his lectures.[33] In May 1932, in his "Philosophy" course, Wittgenstein had turned toward applications of his views to the foundations of logic and mathematics, formulating the need for an explicit uniqueness rule in (quantifier-free) equational specifications of recursive definitions as a replacement for an explicit principle of mathematical induction.[34] Seeking a mathematical audience—and throughout the semester when Turing was reading Russell—Wittgenstein began teaching a new course, "Philosophy for Mathematicians", in the fall of 1932, running it through the spring of 1933. It is possible that Turing attended, although we cannot be sure.

What is even more likely is that, as a rising third-year undergraduate, Turing was aware of controversy about Wittgenstein's views by the fall of 1933. In May 1933 Braithwaite published a survey of Cambridge philosophy since the War, specifically designed to help undergraduates in their choice of a course of study. In this essay he held that "common sense" and "science" are opposed, attributing to Wittgenstein a strong form of verificationism, based on what he knew of his lectures (1933). In a letter to *Mind* Wittgenstein objected heatedly to Braithwaite's characterization of his philosophy, and Braithwaite replied, somewhat testily, that the world awaited Wittgenstein's publication of his views.[35] Perhaps as a result of this, so many students showed up for "Philosophy for Mathematicians" in the fall of 1933 that useful discussion became impossible.

Wittgenstein surprised everyone by cancelling the class and dictating notes to a small group of mathematics students as a substitute for the lectures. Mimeographed and bound, *The Blue Book* (1933–1934) and *The Brown Book* (1934–1935) (1965 (BB))—named for the colors of their covers—were widely distributed, and presumably fairly widely discussed, among Cambridge mathematics students.[36] The so-called *Yellow Book* (a compilation of notes of the lectures given before the cancellation of the course and other dictations 1933–1934) was also dictated, and though perhaps less widely circulated; it gives us a first-hand account of the beginning of the originally envisioned course.[37]

---

[33] McGuinness (2008), pp. 182–187, 207, 256. Cf. Monk (1990), p. 30; McGuinness (1988), pp. 62, 96, 155.

[34] The Lecture was given on May 20, 1932, cf. Stern, Rogers, and Citron eds. (forthcoming); cf. Goodstein (1945), p. 407n, von Plato (2014), Marion and Okada (unpublished).

[35] The letters appeared in *Mind* vol. 42, no. 167 July 1933, pp. 415–16; they are reprinted with editorial commentary in Wittgenstein, Klagge and Nordmann (eds.) (1993), pp. 156–159 and in McGuinness (2008), p. 210.

[36] Wittgenstein dictated *The Blue Book* to Turing's fellow mathematics students H.S.M. Coxeter, R.L. Goodstein, and Francis Skinner, along with Alice Ambrose (writing with Wittgenstein and Newman), as well as Margaret Masterman (later Braithwaite); cf. BB, Preface.

[37] Cf. BB and also the Francis Skinner archives, now at the Wren Library of Trinity College, Cambridge. These were donated by his close friend and colleague (and fellow mathematics student of Turing) R.L. Goodstein to the Mathematical Association of Great Britain, who held them until his death; Wittgenstein had given him the materials soon after Skinner's own death. These contain more extensive and precise dictated material from the period 1933–1935, including a different version of *The Yellow Book* and some hitherto unpublished and unknown lecture notes on self-evidence in logic (cf. Gibson (2010)).

These lectures and dictations mark the first public emergence of Wittgenstein's mature philosophy; the only mimeographed materials that would appear and be widely read in his lifetime. In them we see the issues of "phraseology" and "nonsense" explicitly raised and explored—both notions we shall encounter again in Wittgenstein's and Turing's later writings and discussions. We also see an "anthropological" approach to logic and meaning—due in part to Wittgenstein's discussions with Sraffa[38]—in which various "tribes" are investigated in thought-experiments. Calculations and rules are construed in terms of tables with command lines directing humans to use signs in a step-by-step manner. The problem of how to regard the amalgamazation of such "logic-free" routines is in full view,[39] as is the repeated imaginative circumscribing of human actions, step-by-step, with directive rules, and the difficulty of accurately describing *what* following a rule or set of rules *is* in ordinary English. Perhaps most significantly, Wittgenstein used a series of "language-games" to extrude psychological states from his characterization of rule-following and meaning. What it *is* to be "guided by" a calculative rule is neither a particular mental state nor a specific conscious process. It is something emerging against the backdrop of a series of mergings of routines, and in a shared context of "common sense"—i.e., *sense* held in *common* among us, as we speak, argue, and converse with one another.

5.3.2 "Common Sense" at Cambridge

The points above about Russell, Wittgenstein and Littlewood are indicative. At the center of the tradition Turing enter encountered in 1931, when he first came up to Cambridge (the very year of Gödel's (1931) paper), was the question, "What is the *nature* of the logical?" This was in part a purely philosophical, and in part a mathematical question. First, it raised the issue of how and whether to draw a line between mathematics and logic. Should we take mathematics to be nothing but a branch of logic conceived as a theory of concepts, properties, propositional functions, as Russell claimed? Or is it different in character, though subject to logic, whose essential feature is the symbolic representation of well-founded, step-by-step, iterable formal procedures, some ending in tautologies, as Wittgenstein held? Finally, is mathematics autonomous from logic, rooted ultimately in the "prejudice" and experience of "the mathematician-in-the-street", as Hardy called it (1929 p. 4)? Most importantly, how was one to argue such questions? *Who* was prejudiced, the logician or the mathematician-in-the-street?

[38] Sen (2003), Engelmann (2013a).

[39] BB §41, p. 98. Wittgenstein remained unsatisfied with *The Brown Book*; after an attempted revision in the fall of 1936, it was abandoned. The beginning of Item 152, a notebook begun presumably by early 1937, revisits rule-following, but begins concretely, with a series of calculations from the theory of continued fractions.

Second, this tradition of asking about the *nature* of logic drove logicians themselves back to the foundations, as we have seen above (Sect. 5.2). These, as Wittgenstein would argue, lie in our everyday judgments about what follows from what, what makes sense and what does not, what is convincing and what is not: the human interface with logic.

Littlewood's and Wittgenstein's remarks resonate directly with a longstanding Cambridge tradition of philosophical argument over the notion of "common sense". It was spearheaded by Moore, Keynes and Russell, and developed by Hardy, Ramsey, Sraffa, Wittgenstein, Broad and Braithwaite during Turing's undergraduate years. It concerned philosophy's and logic's relation to what ordinary, mortal, untutored, unspecialized humans might be said to think and mean. May "common sense" be dislodged or changed by a logician or philosopher, or is it instead something to be protected from metaphysical—especially Idealist—onslaught?

For Moore, "common sense" and "plain meanings" were not to be overturned by metaphysics—though Moore himself invoked quite abstract metaphysical ideas in the course of his efforts to refute Idealism and to secure a subject matter for psychology. For Russell, by contrast, untutored "common sense" amounted to nothing more than the primitive, pre-scientific "metaphysics of the Stone Age", and it should be revised, if not overturned, by philosophers responsive to the course of scientific progress.[40] For Ramsey, the important point in discussions of logic had been to find a "common basis" from which to argue with an opponent.[41] For Wittgenstein, the basis of logic was to be achieved only through logic itself; but the key to his post-1930 method was exploration, in informal language, of specific analogies and comparisons used in philosophical discussion of formal and mathematical proof.

The point is that at Cambridge during Turing's undergraduate years the general issue of "common sense" was deeply entrenched, not only in mathematics common room discussions, but in the teaching of logic, philosophy and mathematics. As for the issue of decidability, it was Wittgenstein who had first posed the *Entscheidungsproblem* for general logic (in a letter to Russell of 1913).[42] Ramsey, aiming to develop a foundation for classical mathematics from Wittgenstein's *Tractatus*, contributed a brilliant partial resolution in 1928 (1930) and more general foundational and philosophical discussions in his (1926, 1927, and 1929). Arriving at King's College just 1 year after Ramsey's death, Turing would have been aware of this tradition.

Such considerations also apply to G.H. Hardy, the great number theorist and colleague of Littlewood's who returned to Cambridge from Oxford in 1931, in Turing's

[40] Russell's remark is quoted without attribution in J.L. Austin's "Plea for Excuses" (from 1956, in Austin (1979), p. 185; it was apparently widely known. Wittgenstein sent Russell *The Blue Book*, hoping for a response, in 1935 (McGuinness (2008), p. 250). As Cavell has suggested (2012, pp. 30ff), the opening of Wittgenstein's *Blue Book*, with its truncated, cave-man-like language-game of builders, may be regarded as either a stimulus or a response to Russell's remark about "the metaphysics of the Stone Age".

[41] See Ramsey, (1927), p. 161 and (1929). For an argument that Ramsey's pragmatism influenced Wittgenstein in 1929–30, see Misak (2016).

[42] McGuinness (2008), pp. 56–69; cf. Dreben and Floyd (1991).

first year (Turing had begun reading Hardy's *Course of Pure Mathematics* the semester before he arrived at Cambridge).[43] One of Hardy's main aims in his widely-known paper "Mathematical Proof" (1929) was to urge the study of Hilbertian metamathematics upon Cambridge students of mathematics. He had a sense that, due to *philosophical* objections, the contributions of Hilbert were being overlooked. This required him to criticize the philosophical, though not the mathematical elements of Hilbert's work, the "formalism" that it seemed to contain.

Hardy's whole essay was intended to locate a place where "the-mathematician-in-the-street" could find a haven, safe from both formalist and logicist reductions. By insisting that there is "a [purely mathematical] content given independently of all logic", perhaps "some sort of concrete perceptible basis" of signs, Hilbert and Weyl *precluded*, Hardy argued, any real treatment of mathematical concepts. Hardy (rightly) took Hilbert to believe that there are such concepts, and he also (rightly) took Hilbert's metamathematical investigations of consistency to be mathematically and conceptually important (p. 11). By contrast, Russell, Ramsey and Wittgenstein, the "logisticians" had developed a theory of concepts directly, from logic. This approach Hardy preferred; but at the same time he felt the "logisticians" had wandered too far from the common sense, the "prejudices", of "the-mathematician-in-the-street"—a figure with whom Hardy identified (p. 4).

Hardy made what he took to be a "fatal" objection to the idea of purely mathematical experiences of "strings" and "marks":

> If Hilbert has made the Hilbert mathematics with a particular series of marks on a particular sheet of paper, and I copy them on another sheet, have I made a *new* mathematics? Surely it is *the same* mathematics, and that even if he writes in pencil and I in ink, and his marks are black while mine are red. Surely the Hilbert mathematics must be in some sense something which is common to all such sets of marks. I make this point here, because there are two questions which suggest themselves at once about Hilbert's marks. The first is whether we are studying the physical signs themselves or general formal relations in which they stand, and the second is whether these signs or relations have 'meaning' in the sense in which the symbols of mathematics are usually supposed to have meaning. It seems to me that the two questions are quite distinct (1929, p. 11).

Hardy then invoked Wittgenstein's *Tractatus* theory of "general form". Yet his argument that the ability to copy *the same* sign down is fundamental to the symbolism of mathematics is, after all nothing but "common sense". The idea was granted, developed and discussed later on by Wittgenstein and Turing[44]; it was a problem known and addressed by other logician-mathematicians, such as Post.[45]

---

[43] Hodges (1983/2012), p. 58; cf. the early editions (in 1933, 1938) of Hardy (1941); presumably Wittgenstein read out passages from the 1933 edition in his "Philosophy for Mathematicians" of 1932–1933 and fall 1933. Later Wittgenstein would annotate his copy of Hardy (1941) and copy these annotations into his manuscripts; cf. Floyd and Mühlhölzer (unpublished).

[44] After discussing these issues with Turing, both in 1937 and in his 1939 seminar, Wittgenstein wrote a long series of manuscript pages exploring the idea that a proof must be "surveyable" (*Übersichtlich*), i.e., "can be copied, in the manner of a picture" by a human being, and so "taken in", communicated, archived, recognized, and acted upon; cf. Mühlhölzer (2006), Floyd (2015).

[45] Cf. Post (1941/1994), at n. 8 (p. 377); Post is referring to his initial work on "operational logic" for sequences done in 1924.

Hardy next criticized the "logisticians" for doing away with the idea of a pure "proposition" as a conceptual complex carrying within itself the nature of being true or being false. He argued that *this* idea (Moore's and Russell's, 1899–1910) belonged to "the-mathematician-in-the-street". When he offered a proof in mathematics, Hardy insisted, he ultimately *pointed* at an abstract proposition's intrinsic truth (or falsity), as if pointing at a distant mountaintop, aiming with chains of reasoning to get his pupil also to *see* (p. 18). Hardy conceded that in the end, this analogy was partial, and risked a vicious regress into ineffable pointing, making of proof something merely "gas", an ethereal atmosphere of chatter inessential to the mathematical content at issue. But he "detested" Russell's, Wittgenstein's and Ramsey's rejection of the view, their "multiple relation theory of judgment", which made judging *itself* into a complex ordered fact, and truth into a matter of projection, a greater-than-2-arity structure given by the relations, orderings and objects involved in what is judged.

As Hardy explained, he and Littlewood called "gas" all those dispensable heuristics, "rhetorical flourishes designed to affect psychology, pictures on the board in the lecture, devices to stimulate the imagination of pupils" (1929 p. 18). Thus were philosophy and ordinary language reduced by Hardy—though not by Littlewood—to mere fluff, a trading of opinions and feelings, not something belonging to the *content* of mathematics. Hardy alluded to the *Entscheidungsproblem*:

> The image [of "gas", that which manages to "point" at mathematical propositions as if they are distant peaks] gives us a genuine approximation to the processes of mathematical pedagogy on the one hand, and of mathematical discovery on the other; it is only the very unsophisticated outsider who imagines that mathematicians make discoveries by turning the handle of some miraculous machine. (1929, p. 18).

In this way, using what was ultimately an overstatement, Hardy rejected the idea of reducing all of mathematics to step-by-step calculative routine.

The primary aim of Wittgenstein's 1932–1933 course "Philosophy for Mathematicians" was to offer a more articulate conception of the *positive* role to be played both by calculation and by everyday mathematical language in pedagogy and in mathematical discovery. He was working through Hardy's idea of a "pure" proposition by getting to the "gas".[46] The issue of how we can make sense in ordinary language—how *we* can lay specific conditions down on reality, truly or falsely, how *we* reason at all—had to be accommodated within the foundations of mathematics itself, and not dismissed. For this Wittgenstein explored the ideas of "mechanism" and "calculation" *in* ordinary mathematics and in ordinary language. The idea was to get rid of Hardy's idea of a "miraculous machine" by bringing the figure down to earth.[47] Given Hardy's argumentation, an alternative analysis of metamathematics would be required, as well as an alternative analysis of the very notions of "proposition", "ground" and "conviction". And there would be no way to draw in

[46] On "pure propositions" as gaseous, see AWL, p. 55.

[47] Discussions of a variety of conceptions of "machines" and "mechanism" appear in Wittgenstein's "Philosophy" lectures 1934–1935 and *The Yellow Book*; cf. AWL, pp. 52–53, 72, 80.

and work through the "gas" other than to plough through everyday language, analogies, and "phraseology", mathematical and otherwise.

Wittgenstein's 1932–1933 "Philosophy for Mathematicians" opened with consideration of two major questions, and an answer:

> 1. Is there a substratum on which mathematics rests? Is logic the foundation of mathematics? In my view mathematical logic is simply part of mathematics. Russell's calculus [i.e., *Principia Mathematica*] is not fundamental; it is just another calculus. There is nothing wrong with a science before the foundations are laid. ((1979 (AWL), p. 205).

In answer to Russell and Hardy—as he made explicit in the course of these lectures, by reading out passages from Hardy's *Course of Pure Mathematics* and working through many examples—Wittgenstein was denying that mathematics rested upon any "substratum", logical or otherwise. *Its* foundations were all right. Moreover, *Principia Mathematica*, far from "reducing" mathematics to logic, should be regarded as "just another calculus", simply a *part* of mathematics, and no such "substratum". Wittgenstein would not offer any independent "logical" foundation of mathematics, he was turning instead toward a philosophical "merging" of logic and mathematics, as Turing would later call it (see Sect. 5.7 below)—something different from what either Hilbert or Carnap had envisaged.

Wittgenstein next turned to "a = b", and "a = a", the fundamental steps to be taken in in the grasp of recursive or inductive specifications involving a uniqueness rule. At issue here is the very basis of substitution as a logical step, and the basis of generalizations based on our grasp of step-by-step routines. Where are we to take ourselves to *begin* in grasping the number series? This problem isolated and made vivid a more general and basic difficulty Wittgenstein faced with logical analysis itself, given his own emphasis on well-founded, calculative procedures as the heart of logic and mathematics: What is the starting point of a well-founded procedure, if there are no absolute starting points?[48] How and where does analysis bottom out?[49] This entangled him in the question of what standpoint to use to make sense of the whole idea of *sameness* or *difference* of symbolic steps and substitutions, the "recurrence" of terms in the carrying out of operations.

In his final lecture Wittgenstein sketched some implications of his perspective. First, he positioned himself orthogonally to the reigning "schools" of logicism, formalism and intuitionism. There was some truth in formalism, he held, in that "what counts in mathematics is what is written down", and "if a mathematician exhibits a piece of reasoning one does not inquire about a psychological process"

[48] In the *Tractatus* (1922) there had been posited an ultimate starting point, the "simple objects" of the final analysis. Wittgenstein had rejected these by 1929.

[49] The problem is evinced by his use of images of a spiral to gauge infinitary, rule or law-governed series at Wittgenstein AWL, p. 206. The image had been tethered to a particular origin in *Philosophical Remarks*, Wittgenstein's fellowship submission (1980a) (PR) §§158, 189, 192, 197. But in the "Big Typescript" of 1932–1933 (2005), p. 379, as well as "Philosophy for Mathematicians" 1932–1933, AWL, p. 206 the *issue* of a point of origin and how to assign it in a single space has come to the fore. After this point, the image of the spiral is not used by Wittgenstein, but is replaced with that of a more localized, free-standing and "portable" table, or finite set, of commands.

(AWL, p. 225). Thus an intuitionist "should be asked to show how meaning operates", for it is not enough to insist that the object of mathematical thinking is not signs, but objects. However, Russell was wrong to think that logic is a kind of "zoology" of the conceptual world. And Hardy was wrong to think of mathematical content as lodged in pure conceptual structures at which we can only point, with "gas". So far as his own investigations were concerned,

> There is no retreat in mathematics except in the gaseous part. (You may find that some of mathematics is uninteresting—that Cantor's paradise is not a paradise.) (AWL p. 225)

The point was to revise, not the actual mathematics the students were learning, but rather its mode of presentation, by exploring and comparing the images and heuristic ways of thinking characteristic of the presentation of modern mathematics. "Gas" could and should—indeed *must*—be worked on and discussed. Wittgenstein would admit no sharp break between "gas" or ordinary heuristic discussion and mathematics itself. He thus echoed Littlewood's earlier plea for allowing, in the face of "cantankerous" requests for explicitness, an appeal to plain and ordinary language. He also refused, like Littlewood, to draw a sharp line between mathematics and logic.

Unlike Hardy and Littlewood, Wittgenstein drew a philosophical consequence: by focussing on such fundamental questions as what we are to make of the *first* step in the specification of a class by way of a recursive definition, his students might end up feeling that abstract modern mathematics (e.g. Cantor) is less interesting to them than such foundational questions about the fundamentals. Thus back and forth about how to present mathematical issues, the philosophical "gas" *could* play a serious role in mathematics.

Wittgenstein ended his course this way:

> The talk of mathematicians becomes absurd when they leave mathematics, for example, Hardy's description of mathematics as not being a creation of our minds. He conceived philosophy as a decoration, an atmosphere, around the hard realities of mathematics and science. These disciplines, on the one hand, and philosophy on the other, are thought of as being like the necessities and decoration of a room. Hardy is thinking of philosophical opinions. I conceive of philosophy as an activity of clearing up thought (AWL, p. 225).

Wittgenstein's challenge to Hardy was explicit: philosophy is not a *mere* "decoration", not a *mere* swapping of common room opinions or feelings. The "gaseous" "decorations" in which results are dressed up in what Littlewood called "provisional nonsense" are not only a necessity, they are a serious matter, as much a working part of the communication of mathematics as the "hard" results. So Wittgenstein would again insist in 1939, when he and Turing revisited the discussion of "gas" and the metaphor of "decorations" drawn from Hardy in the very first lecture of his Cambridge course (cf. Sect. 5.6 below).

All this shows that by 1933 at Cambridge a number of mathematics and philosophy students had been exposed to fundamental logical questions about recursive processes, representations of calculational activities, forms of generality, and philosophical questions about the foundations of logic and mathematics. They worked closely with Wittgenstein, Hardy, and Littlewood, each of whom, like Newman, had

published on philosophical issues connected with *Principia*. Each of these figures were to be found in discussion with one another, sometimes in public venues (the Moral Sciences Club, the Trinity Mathematical Society, Wittgenstein's courses), sometimes in the advising of graduate students, sometimes in the common rooms and dining halls of the colleges.

Before we turn to review the theme of "common sense" in Turing's writings, we need to make a few final remarks about Turing and Wittgenstein.

5.3.3 Turing and Wittgenstein

We do not know exactly when Turing first heard Wittgenstein. There is no direct evidence about this in the period of Turing's undergraduate (1931–1934) and early King's College fellowship years (1934–1936). Hodges tells us, however, that they would have known one another during Turing's undergraduate years at the Moral Sciences Club.[50] There was also the Trinity Math Society, where Wittgenstein, Littlewood and Hardy spoke[51] and where Wittgenstein's intimate friend and amenuensis, Francis Skinner—undergraduate 1930–1933, mathematics Fellowship student at the same time as Turing 1933–1936, dictatee of *The Yellow Book*, *The Blue Book* and *The Brown Book*, and other Wittgenstein lectures—became Secretary in 1931.[52] In addition, of course, there was the mathematics common room, and, around the colleges—and at King's specifically—there were many go-betweens with Wittgenstein among the mathematics students who had fellowships during Turing's years.[53]

What we have so far argued, as a partial explanation for, or result of, his turn to Russell in March 1933, is that Turing may well have attended, heard about or read dictations of Wittgenstein's lectures during his second or third year as an undergraduate 1932–1933. We shall further argue that a lecture Turing gave to the Moral Sciences Club in December 1933 lecture is fully consistent with this hypothesis (Sec. 5.4 below). Finally we shall argue that Wittgenstein's idea of "tautology" resonates strikingly with Turing's *specific* method of argumentation in "On Computable Numbers" (see Sect. 5.5), written 1935–1936.

Turing and Wittgenstein certainly held one-on-one discussions about logic well before the 1939 lecture course of Wittgenstein's (on which see Sect. 5.6 below). Hodges tells us that they were introduced in 1937 by Wittgenstein's close student, mathematician and philosopher Alister Watson, a King's College undergraduate

[50] Though it is unclear exactly when, because after 1931 there was a hiatus in Wittgensteinstein's attendance. Cf. Klagge and Nordmann (eds.) (2003), p. 377.

[51] See the archive of talks at https://www.srcf.ucam.org/tms/talks-archive/#earlier.

[52] Klagge and Nordman (eds.) (2003), p. 362.

[53] McGuinness (2008), p. 207 mentions Mary Cartwright, George Temple, L.C. Young, and H.D. Ursell; there were also Goodstein and Skinner, on which see Gibson (2010).

with Turing during Turing's second undergraduate year[54] and, like Turing, afterwards a Fellow of King's College (1933–1939), beginning 2 years ahead of Turing.[55] Watson attended many of Wittgenstein's lectures 1930–1938 (including the 1939 ones) and was close to Wittgenstein.[56] As Hodges also tells us, during the academic year of 1935–1936, just after his election to a King's Fellowship, Turing was observed discussing the nature of Cantor's Diagonal argument with Watson and asking a few questions of Braithwaite.[57]

To Newman Turing apparently did not talk at all about his machines before handing him the manuscript of "On Computable Numbers" in April 1936.[58] Given the unfamiliar character of Turing's argumentation, Newman at first did not know what to make of it, thinking it must be wrong.[59] As we have said, Turing's particular way of resolving the *Entscheidungsproblem* was not the application of a preexisting blueprint of ideas and methods in the metamathematics literature, it was instead a philosophically reflective analytic exercise.

It was thus with philosophers and peers, and not with Newman, that Turing was chatting during his crucial year of conceiving "On Computable Numbers".[60] Moreover, Turing continued philosophical discussions immediately after his great paper was published. By February 1937 he had sent off the first round of offprints of "On Computable Numbers".[61] He reported to his mother that in addition to King's College colleagues he had sent them to—in order—Littlewood, Wittgenstein, Newman and two others. He asked his mother to send one to Russell (warning her not to address him as "Lord").[62] Moreover, when 6 months later Turing returned to Cambridge for an August visit from his sojourn at Princeton, he held discussions

[54] Watson (1908–1982) entered King's College in 1926, receiving firsts in both parts of the Mathematical Tripos; he was awarded Studentship prizes in 1929, 1930 and 1932. Though he failed in his first bid for a King's Fellowship, "Chance and Determinism in Relation to Modern Physics" (1932) he succeeded in 1933, with a new thesis on "The Logic of the Quantum Theory". He helped proofread Braithwaite's edition of Ramsey's papers (1931). A man of the left and a friend of Anthony Blunt, his alleged entanglement with the Cambridge Spy Ring (he never confessed) is discussed in Wright (1987).

[55] Hodges (1983/2012).

[56] Cf. McGuinness (2008), pp. 253, 280.

[57] Hodges (1983/2012), p. 109.

[58] Hodges (1983/2012), p. 109.

[59] Hodges (1983/2012), p. 112.

[60] Hodges (1983/2012), p. 109 tells us that Turing did discuss his ideas about machines with David Champernowne, who, along with Alister Watson, would later be relied on by Sraffa to check the mathematics of Sraffa (1960/1975). See Kurz and Salvadori (2001).We note that Yorick Smythies, Wittgenstein's main amenuensis for his lectures 1938–1941 (cf. Munz and Ritter (eds.) (forthcoming)), took down notes of Max Newman's 1935 logic course (now located at St. John's College, Library, Cambridge). Cf. Copeland (2011), p. 152n3.

[61] Turing was disappointed at receiving only two requests for offprints, one from Braithwaite and one from Heinrich Scholz, who gave a seminar on it at Münster (Hodges (1983/2012), pp. 123–124.

[62] Turing to his mother February 11, 1937 AMT/K/1/54 in the Turing Digital Archive http://www.turingarchive.org/browse.php/K/1/54.

with Watson and Wittgenstein about the implications of his work on Gödelian incompleteness and the *Entscheidungsproblem* for the foundations of mathematics, meeting with them sometimes in the Botanical Gardens.[63]

This discussion occurred at a significant time. Turing was apparently highly optimistic about rewriting the philosophical foundations of analysis. [64] He was deeply involved in writing his Princeton dissertation on ordinal logics (1939), where his (once again) anthropological comparison of higher-order recursion to an "oracle" would be used to explore a procedural analysis of Gödelian incompleteness in terms of higher types (cf. Sect. 5.6 below). Wittgenstein was in the midst of drafting his second book, *Philosophical Investigations*. He lacked the the second half that he had always sought, the application of his mature perspective to logic and the foundations of mathematics. But immediately after talking with Watson and Turing he travelled to Norway and completed the first full draft of his book.[65] His manuscripts commence with a summary of his reactions to the discussions, including remarks on Gödelian incompleteness, and one in which a machine is imagined to be capable of "symbolizing its own action", this clearly indebted to "On Computable Numbers".[66]

Over the next year Wittgenstein would write at length about two alternative ways of looking at Cantor's diagonal argumentation: one bottom-up, in terms of calculative procedures, the other top-down, in terms of theories of cardinality.[67] It is plausible to think that he was directly inspired by Turing's great paper, for in his later 1947 remarks about Turing's "Machines" he not only explicitly reminisced about his 1937 discussions with Turing and Watson, he wrote down a reconstruction of Turing's variation on Cantor's diagonal argument in "On Computable Numbers" (cf. Sect. 5.5 below).[68] For his part Watson published a paper in *Mind* (1938) in which he thanked Turing and Wittgenstein for his understanding of Gödel's theorem.[69]

---

[63] Hodges (1983/2012), p. 136.

[64] Hodges tells us that Turing was "overoptimistic" at this time "in thinking he could re-write the foundations of analysis" ((1999), p. 19). It was just this question that interested Watson in his (1938) paper that they discussed in the summer of 1937 with Wittgenstein; our best guess is that the three urged one another on in conversation to ponder the question. Gandy (1988) p. 82, n26 reports that Turing initially planned a sequel to (1936/7) in which computable analysis would be developed. But he did not proceed, given the fact, pointed out to him by Bernays, that not all real numbers have unique representations as binary decimals (cf. Turing (1937)). Serving as an examiner of Goodstein's PhD thesis in 1938, Turing spotted errors in the work that stemmed from underestimating this problem.

[65] Wittgenstein (2001), "Frühversion"; in (1999) TS 225, 220, 221.

[66] (1999), MS 119, pp. 28ff; cf. PI §190. The first passages Wittgenstein wrote up on Gödelian incompleteness are also then, in Sept 1937 ((1999), 118, p. 106ff; cf. 117, pp. 147, 152; 121, pp. 75v–78v, 81v–84r), and 122, p. 28v; cf. Floyd (2001b) and Floyd and Putnam (2000, 2012) for discussion).

[67] Cf. Wittgenstein (1978) (RFM), Part II, written in 1938, analyzed in Floyd and Mühlhölzer (unpublished).

[68] Wittgenstein (1980b) (RPP I) §1096ff written 1947 and discussed in Floyd (2012b).

[69] Cf. Floyd (2001b).

The fact is that philosophers were among those to whom Turing turned both before and after publishing his great paper.[70] And again: in 1939, back teaching mathematical logic at Cambridge, Turing attended Wittgenstein's lectures on the foundations of mathematics (cf. Sect. 5.6 below).

Let us next turn, one by one, to consideration of the theme "common sense" in a series of Turing's lectures, remarks, and writings.

5.4 Turing's Moral Sciences Club Lecture, December 1933

In December 1933—9 months after reading Russell, at the end of the semester when Wittgenstein was generating *The Yellow Book* and the beginning of *The Blue Book*—the third-year undergraduate Turing read a paper to the Moral Sciences Club on the topic of "Mathematics and Logic". Although all we have as a record is Braithwaite's minutes of the meeting, given what we have just said, it is striking to see what he reports:

> On 1 December 1933, the minutes in the Moral Science Club records: "A.M. Turing read a paper on 'Mathematics and Logic'. He suggested that a purely logistic view of mathematics was inadequate; and that mathematical propositions possessed a variety of interpretations, of which the logistic was merely one." Signed, R.B. Braithwaite (Hodges 1999, p. 6).

These remarks—made to a philosophical audience—raise crucial issues about what may have been Turing's and others' thinking about logic at this time. We shall argue that they may be taken to anticipate the development of a distinctive attitude that Turing would put to practical use later on. If our suggestions are at all plausible, this would show that he had already begun to assemble a distinctive way of thinking about logic while still an undergraduate.

What was meant by "a purely logistic view of mathematics"? And what could have been meant by it an "inadequate" "interpretation" of mathematics?

It is first of all striking that the term "logistic" was used at all.[71] We cannot know whether Turing or only Braithwaite used it. But we do know that Russell, Wittgenstein and Ramsey, the logicists, wouldn't and didn't use it.[72] Hilbert and Ackermann's *Grundzüge der Logik* (1928) used the different term "*Logikkalkül*". Hardy had called Russell, Ramsey and Wittgenstein the "logisticians" while

[70] Of course, conversations with many others went on, especially mathematicians. Newman himself went to Princeton in 1937–1938, joining Turing there, cf. Newman (2010).

[71] A brief history of the term is given in Church (1956), pp. 56–57 n. 125; Church traces the concerted use of "logistic" for mathematical logic back to the 1904 World Congress of philosophy. He notes that "sometimes 'logistic' has been used with special reference to the school of Russell or to the Frege-Russell doctrine that mathematics is a branch of logic", but the "more common usage … attaches no such special meaning to this word". For an overview of *logos, logic* vs. *logistiké* in connection with incompleteness and modern mathematics see Stein (1988).

[72] Except for one loose sheet Wittgenstein inserted into the very end of the manuscript of *Philosophical Investigations* (PI Part II/PPF xiv §372).

surveying their differences with Hilbert's "formalism" and the "intuitionism" of Brouwer and Weyl (1929, pp. 5–6); on this use "logistic" simply meant "symbolic logic". What was meant by Turing's point about "a purely logistic view" would, however, have probably been something else. For by 1933 "logistic" was embroiled, in philosophical discussion, in debates over the *nature* of logic and the *nature* of mathematics.

Turing (or Braithwaite himself) could have picked up the term "logistic" from Lewis and Langford's textbook *Symbolic Logic* (1932), which Braithwaite was just then reviewing (1934).[73] There was interest in the book at Cambridge, as Wisdom, a close student of Wittgenstein's who arrived in 1934, reviewed it in *Mind* (1934). Lewis's earlier *Survey of Symbolic Logic* (1918) may also have been known. Or Cambridge could already have gotten wind of Carnap's *Abriss der Logistik* (1929) or related essays (1931) and his syntax program (1934). Whether or not Braithwaite or Turing knew these or related works and ideas, it is useful to contrast what Turing is reported to have said both with Lewis's and with Carnap's perspectives, reprising our earlier differences with Grattan-Guinness's account.

For Lewis and Langford, "logistic" did *not* mean a particular "view" or "interpretation", but rather a technique or method of approach, roughly coextensive with "symbolic logic". A particular example or "system" of "logistic" would have been something itself already "interpreted" or "applied", the term "logistic" connoting, in general, no specific step of "interpretation" at all, but instead a set of "more or less mechanical operations upon the symbols" (1932, p. 118).[74] In terms of the history of philosophy, since Berkeley "logistic" involved a general algebraic technique of using symbols to fix ideas, reason and calculate—thereby *extruding* vagaries of more ephemeral "ideas" from the mind by means of a kind of substitutive move.[75] Lewis's earlier (1918) *Survey of Symbolic Logic* had actually labelled "heterodox" a conception of logistic and mathematics in which every reference to meaning is eliminated from one's conception of the workings of the formal symbolic system, reducing this to the totality of kinds of symbolic operative order on "strings" of signs. His formulation influenced Post (1936), in his work that would be bound, conceptually, so closely with Turing's, though concocted independently of it.[76]

---

[73] Braithwaite (1934) regards Lewis and Langford (1932) as "eminently successful", predicting that "it will probably not be superseded for some time as the standard work" on the subject of symbolic logic, though he objects to its treatment of the theory of types and finds it wanting as a text in the foundations of mathematics. He notes that Langford's use of postulates is analogous to Hilbert's *Entscheidungsproblem*. Langford, a student of H.M. Sheffer, spent 1925–1926 at Cambridge visiting from Harvard. He met Ramsey and indeed proved some of the earliest significant results on the completeness and decidability of first-order theories, pioneering the use of quantifier elimination (Urquhardt (2012))—another point of contact between Cambridge philosophy and the *Enscheidungsproblem* in the 1920s.

[74] Wisdom (1934), p. 101 rejects this on the ground that these rules "are principles of logic". If Turing had read this, he might have demurred, taking the notion of "calculation" as basic, and, having analyzed *it*, applying it to the whole idea of "logic".

[75] See Detlefsen (2005).

[76] Cf. Mundici and Sieg, Chap. 2 and Davis and Sieg (2015).

In Carnap's *Abriss der Logistik* (1929) and his remarks at Königsberg (1931) the term "logistic" had been explicitly ushered toward a full-throated philosophical perspective, or "view of mathematics". Carnap explicitly construed the term in a narrower, philosophically radicalized, sense. He held that Lewis's conception of "logistic" was oriented toward *extra*-logical applications of symbolic logic (§1) whereas he used it, by contrast, more "purely" (the adjective Braithwaite reported Turing having used), as a philosophical point of view *on* logic and mathematics. This "purely logistic view" was a radical proposal for developing formal systems to analyze philosophical positions. In lectures in London (1934) Carnap would share with an English audience his vision of philosophy as "the logical syntax of the language of science". On this view, the distinction between "analytic" and experiential truths was relativized to particular formal languages, and "every significant question asked by philosophers" was taken to concern, either the logical syntax of a specific formal system or an issue proper to a particular science, thus making the fundamental object of study in logic "direct consequence" and the fundamental problem of philosophy the construction of formal language-systems.[77]

I have no proof that Turing knew in 1933 of any of Carnap's philosophical works, though he did know German, and by 1933 the 1930 Königsberg Conference, where Carnap spoke and Gödel had first announced his incompleteness results, would surely have been heard of and discussed in England.[78] The important point for our purposes, whatever the case, is this. Although in his Moral Sciences Club lecture Turing is said to have invoked the notion of "interpretation", his use of it would have been loose and intuitive, not mathematical. He meant to say that a "purely logistic view of mathematics" is just one among other "interpretations" that purport to tell us what mathematics *really* is, and it is "inadequate" on its own to deliver such a story. By "mathematics" he would have meant mathematics as practiced in ordinary, informal language.

If this is right, and if the gist of Braithwaite's report is accurate, then, like Russell and Wittgenstein and unlike Carnap, in 1933 the undergraduate Turing was surely not ceding the notion of "interpretation" to metamathematics and model theory. After Carnap's assimilation of Tarski's work, "interpretation" would come to be routinely applied by logician-philosophers to the notion of a *structure* or *model* of a formal system, a notion developed by Tarski to clarify the notion of a "definable set". With this notion, collections of sentences may be shown, through an extensionally adequate definition, to be "satisfied ('true') in a model" in a metalanguage, allowing one rigorously to prove the *un*definability of a truth predicate for any sufficiently powerful formalized language within that language and to nevertheless be able to analyze the notion of logical consequence in the system.[79] It is important to stress, however, that notions of meaning, reference and "interpretation" in the

[77] Carnap, Maund and Reeves (1934), p. 47.

[78] Carnap (1931). Carnap himself never once refers to Turing; see Floyd (2012a).

[79] Tarski (1933). Gödel, having studied the Introduction to *Principia Mathematica* and the *Tractatus* carefully, apparently came to the undefinability of truth already in 1930 studying *Principia* directly; Wang (1996), p. 82 and Floyd and Kanamori (2016).

intuitive sense—the one we imagine Turing would have used in his Moral Science Club lecture of 1933—are *replaced* by Tarski in the ascent to a metalanguage, and not analyzed, so that no particular philosophical analysis of meaning or truth follows from Tarski's work.[80] The fact is, we see in Turing's subsequent writings no general philosophical interest in truth *per se*.[81]

Instead, in 1933, in this particular Cambridge *milieu*, I suggest that Turing was positioning himself. He was engaged in a program of putting Russell's new logic into place, critiquing, not only its position or office, but perspectives on mathematics being forwarded by the "purely logistic" views. Turing was insisting that mathematical discourse stand on its own autonomous feet. And he was rejecting the idea that *Principia*—or any other construal of mathematics as a whole in terms of symbolic logic—offers mathematics a "foundation" of any ultimate or privileged sort, or reduces it to logic. A "purely logistic view" is but one (philosophically tendentious) "interpretation" among others, and is *therefore* "inadequate". This is thoroughly consonant with the opening of Wittgenstein's 1932 Cambridge lectures "Philosophy for Mathematicians" we have quoted above: *Principia* is simply part of mathematics, and there is nothing wrong with mathematics before its foundations are laid.

It is true that Turing's words, as reported by Braithwaite, may be taken to express nothing more than a "typical" mathematician's attitude. On this mild view of the remarks, the mathematician should simply regard logic as but one tool in the arsenal, and mathematics as wholly autonomous and self-authenticating. This certainly is consistent with what Turing is reported to have said, and it would echo what Hilbert himself really thought.[82] What is interesting, however, is how directly what Turing is said to have argued resonates with Cambridge philosophical themes central to discussion of his time.

5.5 Turing's "On Computable Numbers" (1936/1937): The "Do-What-You-Do Machine"

Let us turn to Turing' specific argumentation in "On Computable Numbers" (1936/1937), with an eye toward Cambridge resonances. As is well-known, Turing shows that, because of partial recursive functions, it is not possible to diagonalize out of the class of computable functions. As Cantor had demonstrated much earlier, one may diagonalize out of any purported enumeration of the real numbers (or the infinite sequences) by constructing a sequence not on the list: just go down the

[80] Cf. Putnam (2015).

[81] Turing of course duly refers to Tarski in his dissertation on ordinal logics ((1939), p. 197) but here he is interested in *operative* accounts of definability. For discussion of his notion of an "oracle" in (1939), cf. 5.6 below.

[82] Sieg (2013b).

diagonal and change one digit of each expansion.[83] Turing's (1936/1937) applies diagonal argumentation, but differently, as he says explicitly (§8). His particular manner of argumentation adapts Wittgenstein's notion of a tautology, central to the philosophy of logic at Cambridge since the *Tractatus*, as Wittgenstein later explicitly recognized in his 1947 recapitulation of Turing's diagonal argument in terms of "games".[84] Let us summarize the argumentation. We shall dub it Turing's "Do-What-You-Do Machine" argument, for reasons that will become clear.

A Turing Machine is a rigorous mathematical structure, but it is also, from another point of view, a crucial heuristic, or blueprint. It is also, as we have already insisted, an *everyday* picture, a simplified snapshot of human calculation as it is done, boiled down to its simplest and most vivid elements. This second point of view would have counted for Hardy as "gas", a merely rhetorical flourish that somehow nevertheless nonsensically points at a mathematical object. In contrast, for Turing this second aspect of his intuitive model was much more than that. It was an essential part of his making a *philosophically* satisfying analysis of the notion of formal system.

After setting out his analogy with a human computer, Turing gives his definitions. A ***circle-freemachine*** is one that, placed in a particular initial configuration, prints an infinite sequence of 0's and 1's. A ***circular machine*** fails to do this, never writing down more than a finite number of 0 s and 1 s. For Turing the *satisfactory* machines print out infinite sequences of 0's and 1's, whereas the *unsatisfactory* ones "get stuck".[85] A ***computable sequence*** of 0's and 1's is one that can be represented by (is the output of) a circle-free machine. A ***computable number*** is a real number differing by an integer from a number computed by a circle-free machine (i.e., its decimal (binary) expansion will, in the non-integer part, coincide with an infinite series of 0's and 1's printed by some circle-free machine); this is a real number whose decimal (binary) expression is said to be ***calculable by finite means***. Here Turing is applying an intuitive idea, and one incidentally discussed *ad nauseum* by Wittgenstein in his Cambridge lectures (and many other writings), and one also procedurally quite familiar to mathematicians: the notion of a "computable number", conceived as a *rule* for decimal expansion, differs from that of a "real number" conceived *extensionally*, regarded as a finished member of a finished totality.

After presenting examples of machines and variables ranging over machines in the form of "skeleton tables", Turing argues that because there is an enumeration of all the machines (since each can be associated with a "description number"), there is in addition—as a result—a single Universal Turing Machine$U$ that can simulate any Turing Machine.

Next comes Turing's crucial, distinctive argument, his "Application of the Diagonal Process" to show that there is no circle-free machine that enumerates all

[83] Cf. Kanamori (2012a, b).

[84] Wittgenstein RPP I §§1096ff, cf. Floyd (2012a, b).

[85] Watson uses the metaphor that the machine "gets stuck" ((1938), p. 445), but I have not found that metaphor either in Wittgenstein or Turing. In LFM the metaphor is criticized (LFM, pp. 178–179), as well as the idea that we have to fear contradictions more than empty commands.

and only the computable sequences by finite means (§8). His argument turns on a particular and *directly* constructed limit point at which the method itself must end, resolving the *Entscheidungsproblem* in the negative. *Here* is his ultimate reduction of the problem to "common sense".

First Turing defines a hypothetical "decision machine" D, which takes the standard description number k of an arbitrary Turing machine M, and tests to see whether k is the number of a circle-free machine or not, outputting "s" ("satisfactory") if it is, and "u" ("unsatisfactory") if not. He supposes that D computes its enumeration of the description numbers of all and only the circle-free machines by drawing from the enumeration of all machines. Let α_n be the nth computable sequence in this supposed enumeration, and let $\varphi_n(m)$, computable under the hypothesis, be the mth figure in α_n. By combining D with the universal machine$U$, Turing next constructs H, a machine that draws from along the diagonal sequence $\varphi_n(n)$ to enumerate β, the sequence whose nth figure is the output of the nth circle-free machine on input n. Since by hypothesis D is circle-free, so is H.

Turing now argues that there can be no such H, and hence, no such D: these machines may be defined, but they cannot compute computable sequences. He does this by showing that H is infected with a defective command that cannot be followed, viz., "Do-What-You-Do".

The "Do-What-You-Do Machine", H, would by its design enumerate β as follows. Its action is divided into step-by-step sequential sections. In the first $N-1$ sections the integers 1, 2, … $N-1$ have been tested by D, and a certain number of these, say $R(N-1)$, have been marked "s", i.e., are description numbers of circle-free machines. In the Nth section the machine D tests the number N. If N is satisfactory, then $R(N) = 1+ R(N-1)$ and the first $R(N)$ figures of the sequence whose description number is N are calculated. H writes down the $R(N)$th figure of this sequence. This figure will be a figure of β, for it is the output on n of the nth circle-free machine in the enumeration of α_n by finite means that D has been assumed to provide. Otherwise, if N is not satisfactory, then $R(N) = R(N-1)$ and H goes on to the $(N + 1)$th section of its action.

Consider now K, the description number of H itself. We may ask, "What does H do on input K?" Since K is the description number of H, and H is circle-free, the verdict delivered by D on K cannot be "u". But the verdict also cannot be "s". For if it were, H would write down as the Kth digit of β the Kth digit of the sequence computed by the Kth circle-free machine in α_n, namely by H itself. But the *instruction* for H on input K would be: "calculate the first $R(K-1)$ figures computed by the machine with description number K (that is, H) and write down the $R(K)$th". The computation of the first $R(K-1)$ figures would be carried out without trouble. But the instruction for calculating the $R(K)$th figure would amount to "calculate the first $R(K)$ figures computed by H and write down the $R(K)$th". This digit "would never be found", as Turing says. For at the Kth step, it would be "circular", contrary to the verdict "s" and the original assumption that D exists (1936/1937, p. 247). Its instructions at the Kth step amount to the "circular" order "Do What You Do".

The ending of Turing's argument is analogous to drawing a card in a game that says "Do what this card tells you to do". The difficulty is not one of "nonsense" in

the sense of gibberish, or a paradox, or a contradiction forming a kind of barrier to pursuing the hypothesis. The difficulty is that a rule has been formulated, using bits of language we well understand and perfectly acceptable rules of grammar, but which cannot be *followed* without a clear understanding of being in a "certain position" in a game. With a context in which actions are purposeful and directed against a backdrop of well-understood possibilities, "Do what you do" makes perfect sense. Without such a context, it does not. Such is common sense.

The last point matters. Turing's "Do-What-You-Do Machine" argument involves no special appeal to any specific principles of logic, any logical constants, or any interpretation of logic. It does not contain any negation or logical complexity, as most of the other classical uses of diagonal argumentation—including Gödel's—do. It is not a *reduction ad absurdam* argument. Instead, it is comprehensible directly, with—and *only* with—"common sense". Rather than a contradiction, it is the production of this *empty* command—one that cannot be *followed*—that defeats the idea of a decision procedure for logic. The reduction to "common sense" is what makes Turing's resolution so general.

In the usual modern presentation of Turing's proof, one uses the different "Halting argument", due to Martin Davis.[86] In this argument one defines a "contrary" machine C that changes 0 to 1 and vice versa along $\varphi_n(n)$ to enumerate a sequence whose nth figure is the "contrary" output of the nth circle-free machine on input n. When C comes to apply itself to its own description number, it faces a *contradictory* command: "If your output is 0 on this input then you are to output 1, and if your output is 1, then you are to output 0". But no output can both be 0 and 1.

The difference between Turing's and the Halting argument is significant, for Turing's reaches into the "profoundly ordinary". The Halting argument, by contrast, works by *reductio ad absurdum*, applying a contradiction and utilizing the law of excluded middle, just as Gödel's argument for incompleteness did in his famous (1931). Turing, by contrast, does not oblige himself to apply any logical law—of non-contradiction, bi-valence, or the excluded middle—in his proof. Nor is his argument a regress argument, though it can be reconstrued in that way.[87]

Turing's "On Computable Numbers" provides, one might say, a kind of *ex post facto*justification for Wittgenstein's post-1933 "language-game" approach to definability and to logic: a piecemeal, step-by-step exploration of possible meanings, processes, rules, using simplified snapshots of an evolving series of possible routines and modes of argumentation, portrayed as embedded and embodied in an evolving environment and culture. "Logic" is on this view fundamentally erected in the course of the investigation, carried forward in pieces of technology embedded in everyday language, and not by devising a particular formal system or finding any particular bottom level of analysis or uncontroversial starting point.

The notion of *tautology* was Wittgenstein's central contribution to logic and philosophy, as Turing knew. The whole argument of "On Computable Numbers" is a

[86] Martin Davis first gave this argument in 1952; see http://en.wikipedia.org/wiki/Halting_problem#History_of_the_halting_problem and Copeland (2004), p. 40 n 61.

[87] Copeland (2004), p. 38 nicely adapts H to a regress argument.

practical realization—and profound generalization—of an idea Wittgenstein had injected into the heart of Cambridge discussion. The fact that in our language we can put together declarative sayings that obey all the ordinary rules of grammar and yet "cancel out" their saying anything *shows* us something important *in* being empty, something important about logic and its limits. This perspective—as Russell noted at the end of *Introduction to Mathematical Philosophy*—transforms the idea of analytic or self-evident "truth", sinking it into logical activity itself, now regarded as a manifestation of our ability to appreciate limits in the totality of what we can sensibly say, represent, or derive. The role of such limiting cases of expression may be clarified by reflecting on tautologies in a suitable symbolism, just as Wittgenstein had done in the *Tractatus*, and as Turing also did, far more generally and rigorously, in "On Computable Numbers".

5.6 Turing 1939: Wittgenstein's Cambridge Lectures

Space prevents us from offering a detailed interpretation of the most well-known record of Turing's discussions with Wittgenstein (LFM). We shall content ourselves with a brief account of the importance to these exchanges of the theme of "common sense".

The usual focus of interpreters of these lectures has been Wittgenstein's disputes with Turing over the role of contradictions and paradoxes in logic. Wittgenstein says that such contradictions (such as the Liar paradox) are "useless", and he questions why anyone would ever has worried about them (LFM, pp. 207ff). Turing puts up a spirited defense, insisting that contradictions can have real-world consequences. (As a matter of fact, he was already working at Bletchley Park, where he would help design machines to decode cyphers from the German Enigma machine, and part of the implementation of logic used was to knock out possible interpretive hypotheses by finding contradictions.)

Commentators have tended to extract two points from these exchanges, each of which we shall dispute. First, there is the claim that Wittgenstein is attacking mathematical logic, pitting it against philosophy, so that the whole discussion is a pro vs. con debate, Turing pro mathematical logic and Wittgenstein con.[88] Second, there is the idea that Turing is some kind of Platonist, and Wittgenstein some kind of constructivist, perhaps even rejecting use of negation in infinite contexts.[89]

The fact is that both thinkers were primarily interested, instead, in the workings of "common sense" and ordinary "phraseology". From the beginning of the very first lecture, they were picking up where they were before in discussion. Wittgenstein knew that "On Computable Numbers" had made a profound contribution to our understanding of logic; he was interested in developing his own philosophical thoughts to see how far he could better defend them before a sophisticated audience.

[88] Monk (1990), pp. 419–420.

[89] Turing himself worries about constructivism creeping in: cf.LFM, pp. 31, 67,105.

Sometime in 1938 he had given a lecture to his circle of students (including Watson) in which he attempted, somewhat lamely, to transmute the "Do-What-You-Do-Machine" argument of Turing into the language of Gödel's 1931 paper.[90] This brainstorming gave way, in the spring 1939 lectures, to a businesslike reworking of his own philosophical ideas about common sense with Turing in the audience. Gödel's incompleteness theorem is only briefly mentioned (LFM, pp. 56, 188–189), presumably because they had already discussed it before; however certain Gödelian themes, refracted through Turing's (1939) work, shine through in their exchanges, as we shall explain.[91]

The fundamental themes under discussion concerned "gas" and the role of calculations, rules, and concept-formation in mathematics—ultimately, as we have seen from Hardy onward, the issue of how to regard "ground" and "conviction" in logic (cf. 1929, p. 17). Turing and Wittgenstein agreed that the "ground" would not be an object language leading to "an infinite hierarchy", as Wittgenstein put it (LFM, p. 14): neither advocated Hardy's idea of a purely abstract proposition. The "ground", instead, would be methods of procedure *as they are used*, of which, as Turing had himself shown in "On Computable Numbers", there are a variety. Wittgenstein's main aim was to stress how subtle or dramatic conceptual shifts may happen, as reflected in phraseology, when one kind of method gives way to a wholly different one, as opposed to being smoothly amalgamated into a prior whole. Turing repeatedly questioned, not Wittgenstein's views, but his applications of arguments to specific cases. Their most heated exchanges took place around the idea of "common sense".

The trouble with Hardy's idea of "gas" is that it made of such conversation as theirs something merely "aesthetic". This for Wittgenstein was a wholly wrong point of view of the relation of logic and philosophy to mathematics. ("You smoke cigarettes every now and then and work. But if you said your work was smoking cigarettes, the whole picture would be different" (LFM, p. 16)). Wittgenstein's basic move was to insist that there is no *general* dichotomy between form and content, between aesthetic or heuristic and result, between formal procedure and meaning, between philosophical discussion of mathematics and mathematics, and between previously given understandings and their revision—though of course there are *local* distinctions that may be drawn in the face of particular cases, and which particular "phraseology" we choose, and how we respond and go on from it, often very much matters. Wittgenstein wanted to argue that this followed from Turing's own analysis of the very idea of a formal system, in the form of a problem about what it *is* to follow a rule, correctly or incorrectly.

The trouble with the notion of "common sense" as an unexamined idea is philosophical. How are we to distinguish erroneous dogma, or mere stipulation, from common understanding that provides us with sufficient backdrop for the notions of correctness and incorrectness to have a grip? How are we to understand the notion

[90] This, at least, is one guess as to the contents of the lecture, which exists only in notes taken down by Smythies. See Munz and Ritter (eds.) (forthcoming).

[91] For more on Wittgenstein's own remarks on Gödel and "phraseology", see Floyd (2001b).

of the "reasonable", as opposed to the formally possible? How are we to correct—or even just expose and probe—what is taken to *be* "common sense" by a mathematician or philosopher? How are we to make heuristic discussion and the metaphors used something other than "decoration", "a lot of jaw", "like squiggles on the wall of a room", as Hardy had said (1929, pp. 13–14).

We have already seen that this had long been the main divide between the Littlewood-Wittgenstein-Turing Cambridge approach, on the one hand, and the Hardy approach on the other. In *The Blue Book* and *The Brown* Book, as well as in his 1932–1933 lectures and dictations for mathematicians, Wittgenstein had developed a method of thinking through a series of small variations and comparisons in the backdrop of imagined, unformalized human uses of logic ("language-games") to draw out the critical importance, not only of "common sense", but of the kind of conversation and discussion that a reasonable notion of "common sense" requires. Here, in 1939, he applies the method to well-known classical impossibility and other proofs in mathematics.

Wittgenstein's overall suggestion to Turing was to express a concern. Without the right orientation toward his machines, Turing risked being seen, or seeing himself, as nothing but a reducer of mathematics to logic, and logic to "squiggles" (LFM, p. 14). The first step here—just where Wittgenstein began Lecture I—was to insist that a key constraint on his remarks was "not to interfere with the mathematicians" (LFM, p. 13). That is, to say, he wanted to make it clear that he had no interest in revising logic, or Turing's mathematical work. In fact Turing's remarks were of special importance to Wittgenstein, and he even chastised himself later on in the lectures when he appeared to swerve dangerously close to interfering with ongoing mathematics in discussing the notion of contradiction (LFM, p. 223). He wanted to keep his conversation with Turing going, pressing the importance of not covering up the variety of methods at work in logic and mathematics. And Turing represented, within the context of the course, "common sense" as it was understood by mathematicians.

The main question under discussion was how to accept what Turing had done in analyzing the notion of a formal system, and yet draw out its philosophical significance. Lecture I begins with an insider's joke directed at Turing: a kind of tribute. First, Wittgenstein reverted to his extrusion of mental states from the foundations of logic, pointing out that since an expression has different kinds of use, it makes no sense to think that one can "have the use before one's mind" in a single instant. What one has is a *modus operandi*, and not an image-in-itself, or a state of mind as a property or ontological particular. This he argued for by reverting to the older argument—initiated by Hardy in response to Hilbert, as we have seen—about *sameness* of signs (LFM, p. 20):

> What is a 'representative piece of the application'? Take the following example. Suppose I say to Turing, "This is the Greek letter sigma", pointing to the sign σ. Then when I say, "Show me a Greek sigma in this book", he cuts out the sign I showed him and puts it in the book.

This is obviously a parody of the idea of a sign as used by a Turing Machine. Wittgenstein continues:

> —Actually these things don't happen. These misunderstandings only immensely rarely arise—although my words might have been taken either way. This is because we have all been trained from childhood to use such phrases as "This is the letter so-and-so" in one way rather than another.
>
> When I said to Turing, "This is the Greek sigma", did he get the wrong picture? No, he got the right picture. But he didn't understand the application (LFM, p. 20).

The point to Turing was the importance of "taking a wider look 'round", at the general *cultural* and "application" setting within which a machine or routine is set to work. This might be said to be a "common sense" remark *about* "common sense", as well as about computations themselves. But Wittgenstein's point was that the philosopher is duty-bound to insist on the importance and complexity of this notion, for only in this way may meaning, and actual logic itself, get a grip. Attending to all the panoply of apparent contingencies and "phraseology" surrounding our uses of logic, and discussing them one-by-one in turn, learning to contrast and compare cases, is the only way to rightly—or responsibly—see what it *is* to be "responsible to a mathematical reality" (LFM, p. 240).[92]

Further on in the lectures (LFM, p. 35), the issue of how time may or may not enter into an analysis of proof is discussed, a significant feature of Turing's model, for it has a double face: the first that of a static mathematical object or formalism, the other that of a dynamic machine whose movements are traceable by mechanisms and human step-by-step computations as we regard them ordinarily. Wittgenstein explores with Turing the question of enormously long computations and proofs, whose outcomes constantly change (LFM, pp. 37, 105–106), how to think about the relation between formalized and unformalized proofs and the notion of rigor (LFM, pp. 127–133, 261ff), how to demarcate the realm of calculation from that of experiment (LFM, pp. 96ff), how to discuss certainty vs. skepticism about calculated or proved results (LFM, pp. 101–103). All this talk of "faces" and length-of-proof would have been counted by Hardy as "gas", but to Wittgenstein and Turing it forms a crucial matter for discussion, a way of thinking about what it is to "build new roads" in mathematics, as opposed to ineffably pointing at distant mountaintops (LFM, p. 139).

This whole issue of "phraseology" also formed the key to their debates over contradictions. Turing took Wittgenstein at times to be insisting that when a conceptual shift occurs, when the character or aspect of use shifts, there is a "change in meaning", and he repeatedly worried that Wittgenstein was attacking the use of negation or indirect argument in general, as an intuitionist or finitist would—something Wittgenstein explicitly and repeatedly denied (LFM, pp. 31, 67). Turing also worried that Wittgenstein was veering too close to saying that mathematicians merely "invent" procedures, rather than "discovering" them—again something Wittgenstein denied (LFM, p. 68). Instead, Wittgenstein wanted to discuss what

[92] Cf. Diamond (1996) for a defense of Wittgenstein's "realism".

discoveries in mathematics *are*. He reminded the students, after an exchange with Turing, of his "slogan":

> Don't treat your common sense like an umbrella. When you come into a room to philosophize, don't leave it outside but bring it in with you (LFM, p. 68).

Turing challenged this response, especially applied to the arising of possible contradictions: "You seem", he later remarked, "to be saying that if one uses a little common sense, one will not get into trouble" (LFM, p. 219). Wittgenstein responded vehemently,

> No, that is *NOT* what I mean at all. — the trouble described is something you get into if you apply the calculation in a way that leads to something breaking. This you can do with *any* calculation, contradiction or no contradiction (LFM, p. 219).

The issue reaches back to Russell's *Introduction to Mathematical Philosophy*, and the question whether appealing to the laws of non-contradiction, bivalence, and excluded middle suffice for the foundations of logic. For Wittgenstein, as for Russell, they did not. Wittgenstein had long argued that instead, we need to attend to *tautologies*, emptinesses of words and rule-commands, in order to probe the limits of the logical, and see how it is grounded, not in facts or self-evident principles, but in our own manner of representing the world in language by doing things *with* language. This, as we have seen, was confirmed by the "Do-What-You-Do" machine Turing had constructed in "On Computable Numbers"; even there the argument was explicitly portrayed as a response to the kind of worries about negation Wittgenstein was still expressing in 1939.[93]

Turing and Wittgenstein agreed that the use of mathematical symbols in extra-mathematical contexts is crucial to their meaning; such would anyone having learned from logicism.[94] Wittgenstein, however, worried that this could leave parts of mathematics out of account. For some of these (e.g. set theory) he regarded as taking up, as their very *raison d'être*, a way of regarding concepts that abstracts from all the procedures and methods with which we are familiar from those parts of mathematics that are applied (LFM, p. 29, 102–103). Moreover, Wittgenstein rejected Turing's suggestion that the relation between everyday language and mathematical symbolism is merely one of "abbreviation" or "definition" as Turing at one point suggested (cf. LFM, p. 42). *Conceptual*, not merely verbal work is required to

---

[93] In (1936/1937) §8 Turing writes, of the more "direct proof" using a *reductio*, that "although [it is] perfectly sound, [it] has the disadvantage that it may leave the reader with a feeling that 'there must be something wrong'". His "Do-What-You-Do" argument offers a response to such a reader. In his corrections (1937), stimulated by Bernays, Turing develops this point with respect to intuitionism explicitly. See Floyd (2012b) for a discussion.

[94] Russell (1919), p. 6:

> This point, that "0" and number and "successor" cannot be defined by means of Peano"s five axioms, but must be independently understood, is important. We want our numbers not merely to verify mathematical formula, but to apply in the right way to common objects. We want to have ten fingers and two eyes and one nose. A system in which "1" meant 100, and "2" meant 102, and so on, might be all right for pure mathematics, but would not suit daily life.

make unsurveyable routines surveyable, and this is a matter of proof in mathematics, not merely experiment or psychology (LFM, p. 226). The relation between logic and mathematics was to be found in the concepts, but this was not properly clarified by Frege and by Russell, with their "theory of types".

Wittgenstein eventually acknowledged that he wanted to say "something rather similar" about "common sense" to what Turing had suggested (LFM, p. 223). "Common sense" had "some truth" in it (LFM, p. 229). His idea was that the basis and the necessities of logic lie in what we *do*, in what we *recognize* in practice, rather than in our being convinced of a particular truth (LFM, p. 230). Our *uses* of the law of contradiction in particular venues and contexts shows us what adherence to it *is*, and this is constituted, not only by the myriad ways we have been trained and train others in a variety of methods, but also by what we are inclined to say: how we are inclined to express ourselves, draw analogies, pictures, and so on, in conversations *about* particular cases—the "gas".

On this view, the law of contradiction is not to be understood as a stopping point of argument, a potential flaw in engineering routines, or a piece of psychological reality. Rather, we must see the relation of logic to mathematics in terms of what we *grant*, normatively, what we *recognize as*: reasoning, calculating, inferring, and so on. Here Wittgenstein did not simply mean, as Hardy had, our ability to recognize one sign as the same as another. He meant the more general, older notion of recognition, the sense in which a king *recognizes* or *acknowledges* the dominion of a lord over a land. Such recognition, or granting of authority is bound up in serious ways, not simply with the law of contradiction as applied in physics and mathematics, but with how we *phrase* ourselves about hard and transition cases, with what we are able actually to share, with "common sense" in the sense of a working harmony and agreement. This was the domain, at least for Wittgenstein, of philosophy. It is far more than "smoke" or "gas".

Of course, Turing was a mathematician, not a philosopher. His job was to do something mathematical. In the winter of 1937, back at Princeton after his summer discussion group with Wittgenstein and Watson, he had written his dissertation, devoted to the question of how, logically, one might get around Gödelian incompleteness operationally, in theory (1939). He framed a notion of *relative* computability or solubility in a higher-order logic, invoking the picturesque idea of an "oracle": the instantaneous, black-box delivery of a solution otherwise unavailable, getting one to a next system. This anthropological image—one that eliminates the internal psychology of the individual human mind entirely from the mathematical step—was, in the words of Solovay, to "change the face of recursion theory".[95]

It was this very image of an "oracle" to which Wittgenstein—probably aware of Turing's (1939)—explored in his 1939 lectures:

> [*Wittgenstein*]: So what about our case: this new calculation and these people disagree [half go one way, half go the other[96]]. What are we to say?—Shall we say, "Why aren't our minds

[95] Solovay (1988), p. 126; cf. discussion of Turing (1939) in Copeland (2004) and Hodges (1999).

[96] Our best guess is that Wittgenstein is alluding to the situation after Gödel 1931, in which both the Gödel sentence P and its negation, not-P, may each be added to the original system of arithmetic

> stronger?" or "Where is an oracle?" But is there anything for it to know? Aren't you right—or wrong—as you please?
>
> *Turing*: We'd better make up our minds what we want to do.
>
> *Wittgenstein*: Then it isn't a message from God or an intuition, which you pray for, but it is a *decision* you want. But doesn't that contradict [your] idea of an experiment? Where is the experiment now?
>
> *Turing*: I should probably only speak of an experiment where there is agreement.
>
> *Wittgenstein*: Don't you mean that in that case the experiment will show what the rule is? The fact is that we all multiply in the same way—that actually there are no difficulties about multiplication. If I ask Wisdom to write out a multiplication and get the result, and he tells me, then I am perfectly certain that that's the right thing, the adopted thing (LFM, p. 109).

Wittgenstein is here arguing *against* the idea, at first casually suggested by Turing, that different understandings of particular phraseologies represent nothing but different "decisions", "intuitions", or "experiments" (LFM, pp. 109ff, cf. 31, 147).[97] "I should say," Wittgenstein replied, "that if it was a mathematical proof, God didn't know more than any one of us what the result of the calculation was" (LFM, p. 103). His point would have been that Turing's notion of computation relative to an oracle is *not* the idea of a computation less certain than any other mathematical step, and it need not be regarded as something analogous to a proposition of physics, or an approximation.[98] And so it is with a *relative* computation in Turing's (1939) sense. If we take up the result of a higher-order computation, then we take *it* as it is. But it is *we* who acknowledge the result of the "oracle", and we proceed conditionally, as logic and mathematics allow us to do.[99]

What Turing went on to do in the 1940s with Wittgenstein's notion of "phraseology" is what we shall discuss next.

5.7 Turing's "The Reform of Mathematical Notation and Phraseology" (1944/1945)

As we have seen, to Turing the ordinary talk or "common sense" of what Hardy (1929) called the "mathematician-in-the-street" is a richly rewarding subject of scrutiny, as well as a repository of longstanding mathematical experience and culture. It was not *just* "gas". As he saw it, it should be respected, exploited, improved, and given its due. On the other hand, as he lamented in an unpublished paper on

---

consistently, presenting a kind of branch of possible paths. Cf. Wittgenstein's 1938 lecture on Gödel in Munz and Ritter (eds). (forthcoming).

[97] Turing works through the notion of "intuition" himself in (1939), in (2001) §11, pp. 214–216. Cf. Solovay (1988) and Copeland (2004) for discussion of the 1938 thesis, the basis for (1939).

[98] It is just this idea to which Wittgenstein reverts when, probably reminiscing about these exchanges, he revisits the notion of an "oracle" much later. See RPP I §817 and (1974) (OC) §609, written just six days before his death.

[99] In (1944) Post criticized Turing for his "picturesque" use of the idea of an "oracle", writing "the 'if' of mathematics is … more conductive to the development of a theory", p. 311 n.23.

"The Reform of Mathematical Notation and Phraseology" (1944/1945), the variegation of this talk remained "exceedingly unsystematic", constituting "a definite handicap both to the would-be-learner and to the writer who is unable to express ideas because the necessary notation for expressing them is not widely known" (1944/1945, p. 215). He was calling for a "reform" of mathematical notation and "phraseology". As he acknowledged here, he drew his ideas from Wittgenstein:

> We are taught that the theory of types is necessary for the avoidance of paradoxes, but we are not usually taught how to work the theory of types into our day-to-day mathematics: rather we are encouraged to think that it is of no practical importance for anything but symbolic logic. This has a most unfortunate psychological effect. We tend to suspect the soundness of our arguments all the time because we do not know whether we are respecting the theory of types or not. Actually it is not difficult to put the theory of types into a form in which it can be used by the mathematician-in-the-street without having to study symbolic logic, much less use it. The statement of the type principle given below was suggested by lectures of Wittgenstein, but its shortcomings should not be laid at his door (1944/1945, p. 217)[100]

In the essay Turing appeals to argumentation he and Wittgenstein had discussed in Wittgenstein's seminar:

> The type principle is effectively taken care of in ordinary language by the fact that there are nouns as well as adjectives. We can make the statement 'All horses are four-legged', which can be verified by examination of every horse, at any rate if there only a finite number of them. If however we try to use words like 'thing' or 'thing whatever' trouble begins. Suppose we understand 'thing' to include everything whatever, books, cats, men, women, thoughts, functions of men with cats as values, numbers, matrices, classes of classes, procedures, propositions,. .. Under these circumstances what can we make of the statement 'All things are not prime multiples of 6'? We are of course inclined to maintain that it is true, but that is merely a form of prejudice. What do we mean by it? Under no circumstances is the number of things to be examined finite. It may be that some meaning can be given to statements of this kind, but for the present we do not know of any. In effect then the theory of types requires us to refrain from the use of such nouns as 'thing', 'object' etc., which are intended to convey the idea 'anything whatever' (1944/1945, p. 218).

Turing's view, as expressed in this paper, was that "it has long been recognized that mathematics and logic are virtually the same and that they may be expected to merge imperceptibly into one another", although "this merging process has not gone at all far, and mathematics has profited very little [so far] from researches in symbolic logic" (1944/1945, p. 245). Though Turing was nodding to what he called the "Russellian *Weltanschauung*" of *Principia Mathematica*—whose theory of types he took to be held by "the majority of mathematicians-in-the-street"—he was not arguing for logicism as a conceptual or metaphysical doctrine in the foundations of mathematics. He was making an observation. Turing's aim was to revise both symbolic logic and mathematics to effect a better "merging", where this would be a matter of degree rather than a matter of either a sharply "perceptible" line or a reduction of one subject to the other.

[100] Cf. Floyd (2013). Turing's explorations of the logical tradition (Leibniz, Boole, Peano, etc.) continue in a notebook from c. 1942, "Notes on Notations". See https://www.bonhams.com/auctions/22795/lot/1/.

His diagnosis of the difficulty hindering productive "merging" was one of culture and missed opportunities for cooperative communication, rather than any in-principle philosophical mistake:

> The chief reasons for this [merging not going at all far] seem to be a lack of liaison between the logician and the mathematician-in-the-street. Symbolic logic is a very alarming mouthful for most mathematicians, and the logicians are not very much interested in making it more palatable. It seems however that symbolic logic has a number of small lessons for the mathematician which may be taught without it being necessary for him to learn very much of symbolic logic.
>
> In particular it seems that symbolic logic will help the mathematicians to improve their notation and phraseology…By notation I do not of course refer to such trivial questions as whether pressure should be denoted by *p* or P, but deeper ones such as whether we should say 'the function *f*(*z*) of *z*' or 'the function *f*' (1944/1945, p. 215).

Whether to refer to letters of the alphabet, parameters such as "*z*", casually, mixing what Carnap called "formal" and the "material" modes of speech, or whether to objectualize functions; whether rigorously to obey use vs. mention and/or to avail oneself of Russell's theory of incomplete symbols and types, whether to explicitly bind all variables or leave them free, but typed: these issues were on Turing's mind at the time he wrote this essay. Such issues lay at the heart—the philosophical heart—of "symbolic logic". Just here Turing took a particular stand:

> It would not be advisable to let the reform take the form of a cast-iron logical system into which all the mathematics of the future are to be expressed. No democratic mathematical community would stand for such an idea, nor would it be desirable. Instead one must put forward a number of definite small suggestions for improvement, each backed up by good argument and examples. It should be possible for each suggestion to be adopted singly. Under these circumstances one may hope that some of the suggestions will be adopted in one quarter or another, and that the use of all will spread (1944/1945, p. 215).

On Turing's view, formalisms should never be "cast-iron" straightjackets, but opportunistic, targeted aids to the avoidance of ambiguity and unclarity, bridging the gap between the formal and the informal in order to lessen the need for detailed knowledge of formalized structures. Rather than machines for calculating necessary and sufficient conditions, they would be operational, useful interventions. They would respond to the language of the "mathematician-in-the-street", clarifying and aiding it. Turing advised that we make a study of "current mathematical and physical and engineering books and papers with a view to listing all commonly used forms of notation", and then examine the notations

> …to discover what they really mean. This will usually involve statements of various implicit understandings as between writer and reader, it may also include the equivalent of the notation in question in a standard notation…[Then proceed in] laying down a code of minimum requirements for desirable notations. These requirements should be exceedingly mild (1944/1945 p. 215).

The meanings involved in mathematics and mathematical talk are various, but for Turing they are dynamic, evolving, and purposive, subject to notational clarification and systematization in the context of ongoing cultural and intellectual developments. His envisioned "code of minimum requirements" was to serve communication

and clarity, not machine implementation alone. It was certainly not intended to fix a "cast-iron" semantics for our language—one provably unavailable anyway, on his view, due to incompleteness and undecidability.

Turing's paper was written after he had accomplished much work at Bletchley Park. It is forward-looking in its call for the development of what computer scientists nowadays call "types" in higher-level programming languages: categories and ontologies that humans can use, develop, visualize, organize and communicate with, among one another and utilizing machines. Even today the "semantic web" involves a quest for massive archival sorting, exploitation, higher-type organization.[101] Turing never lost sight of the need for articulation in higher-level languages and meanings in his implementation of the very low-level, step-by-step computational and representational modularity of the Turing Machine. He saw the difference in levels and types as a complex series of systematizations sensitive to everyday "phraseology" and common sense, not a divide of principle. This was because he always saw "types" or "levels" as lying on an evolving continuum, shaped by practical aspects, the user end, *and* mathematics. *This* is the aspect of his thought that drew from the Cambridge tradition of "common sense".

5.8 Turing's "Solveable and Unsolveable Problems" (1954)

This brings us to the final published words of Turing, the closing paragraph of his last paper, a popular presentation of logical theory in terms of word problems and puzzles. Here Turing adapts some of Post's work on word problems to this accessible setting, revisiting issues of undecidability and incompleteness.[102] He closes the essay this way:

> These [limitative] results, and some other results of mathematical logic may be regarded as going some way towards a demonstration, within mathematics itself, of the inadequacy of 'reason' unsupported by common sense (1954, p. 23).

Turing's idea of "common sense" is the idea of something *not* given in an algorithm, but generally shared among language users, and ineradicable in light of incompleteness. His stance implies that the reality of what we do and say with our words is part of reality itself, not a mandatory "add on" at "the meta-level" to a purely formalized or idealized computational "object language". The mathematician-in-the-street should not be regarded by the logician as someone with metaphysical prejudices to be argued with, for or against (as Hardy thought), but rather as someone in need of practical help. Help would come from inside mathematics itself, but only with appropriate philosophical understanding.

As Turing's student Gandy put it,

[101] Cf. Wolfram (2013), a commentary on Turing's (1944/1945).

[102] See Davis and Sieg (2015).

> Turing was first and foremost a mathematician. He believed that the chief purpose of mathematical logic and the study of the foundations of mathematics was to help mathematicians to understand what they were doing, and could do. In pursuit of this goal, mathematical logicians must perforce construct and manipulate complex formal systems. But they have a duty to explain to mathematicians, in as non-technical way as possible, what they have accomplished. (A good example is Turing's account of the purpose of his ordinal logics—see his paper [(1939] in Part I). Turing disliked those high priests of logic who sought (like Quine in his 'Mathematical Logic' [1940, 1st. ed.]) to blind the mathematician-in-the-street with arcane formalisms (Turing (2001), p. 213).

Quine's emphasis on syntactic *finesse* and the use of formalized notation to enunciate a view of the world as a whole stemmed from the Russellian and Carnapian traditions. Turing set his face against these. These philosophies of logic were oriented *cosmically*, toward an explicit enunciation or renunciation of an ontology of the world as a whole, and the articulation of meaning through logical consequence, implemented, especially in the hands of Quine and Carnap, through emphasis on syntax. Turing's work stemmed from a different quarter. He was an artful and practical dodger in matters of ontology and meaning, oriented toward the values of use and simplicity all the way down.

5.9 Concluding Remark

This essay is intended a contribution to our understanding of how philosophical progress takes place. Genuine philosophical contributions do not always come labelled with the term "philosophy", and progress in philosophy does not always come labelled with the term "philosophical progress". Sometimes the most crucial moves in philosophy are not labelled as such *per se*, being pitch and not wind-up. In such cases, when what is to count as "common sense" is at stake, what is required is for participants to articulate rather than merely expound, to inspire and adapt ways of thinking, to pursue and cultivate open-minded reflection, argumentation, a sense of internal criticism and careful development of particular, differing points of view. From this may emerge the true value of intelligibility, directness and simplicity—especially at the initial steps.

Such are the earmarks of Turing's philosophical and scientific sensibility. They are to be celebrated and prized.

Acknowledgments I am grateful to S. Barry Cooper, Akihiro Kanamori, and Mauro Engelmann for stimulating conversations about the ideas in this paper, as well as the audience at our Turing 100 conference at BU, November 11, 2012. In addition, the Stanhill Foundation provided me with generous funds for a visit to the Skinner archives at Trinity College Cambridge in June 2015. There, with the able and generous aid of Jonathan Smith, I was able to do a great deal of helpful and inspiring research. Thanks are due to Arthur Gibson, Ilyas Kahn, Susan Edwards McKie, and Jonathan Smith for their conversation during this visit. In the final stages, I also received very helpful feedback from Juliette Kennedy and Adriana Renero.

References[103]

Austin, J.L. 1979. Philosophical Papers. Edited by J.O. Urmson and G.J. Warnock. Oxford/New York: Oxford University Press.

Braithwaite, R.B. 1934. Review of C.I. Lewis and C.H. Langford, *Symbolic Logic* (1932). *The Mathematical Gazette* 18(227): 57–58.

Carnap, R. 1929. *Abriss der Logistik Mit Besonderer Berücksichtigung der Relationstheorie und Ihrer Anwendungen.* Vienna, Julius Springer. English translation by Dirk Schlimm to appear in *Collected Works of Rudolf Carnap, Vol. III: Pre-Syntax Logic 1927–1934*, edited by Erich Reck, Georg Schiemer, and Dirk Schlimm.

———. 1931. Die logizistische Grundlegung der Mathematik. *Erkenntnis* 2 (2/3): 91–105.

———. 1934. *Logische Syntax Der Sprache*. Vienna: Julius Springer.

———. 1935. *Philosophy and Logical Syntax*. London: Kegan Paul, Trench, Trubner & Co.

———. 2000. *Untersuchungen zur Allgemeinen Axiomatik*. Edited from unpublished manuscript by T. Bonk and J. Mosterín. Darmstadt: Wissenschftliche Buchgesellschaft.

Carnap, R., C.A.M. Maund, and J.W. Reeves. 1934. Report of Lectures on Philosophy and Logical Syntax, Delivered on 8, 10 and 12 October at Bedford College in the University of London, by Professor Rudolf Carnap. *Analysis* 2(3), December: 42–48.

Cavell, S. 2012. Philosophy as the Education of Grownups. In *Stanley Cavell and the Education of Grownups*, eds. P. Standish and N. Saito, 1–33. New York: Fordham University Press.

Church, A. 1956. *Introduction to Mathematical Logic*. Princeton: Princeton University Press.

Cooper, B.S., and J. van Leeuven (eds). 2013. *Alan Turing – His Work and Impact*. Amsterdam/Boston: Elsevier Science.

Copeland, J. 2004. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life*. Oxford: Oxford University, Clarendon Press.

Copeland, B.J.. 2011. From the *Entscheidungsproblem* to the Personal Computer—and Beyond. In *Kurt Gödel and the Foundations of Mathematics: Horizons of Truth*, eds. M. Baaz, C. H. Papadimitriou, H. Putnam, D. S. Scott and C. L. Harper, Jr., 151–184. Cambridge: Cambridge University Press.

Copeland, B. J., and O. Shagrir. 2013. Turing versus Gödel on Computability and the Mind. In eds. Copeland, Posy, and Shagrir, 1–34.

Copeland, B.J., C.J. Posy, and O. Shagrir (eds.) 2013. *Computability: Turing, Gödel, Church and Beyond*. Cambridge, MA: MIT Press.

Davis, M. 1982. Why Gödel Didn't Have Church's Thesis. *Information and Control* 54: 3–24.

Davis, M., and W. Sieg. 2015. Conceptual Confluence in 1936: Post and Turing. In *Alan Turing: The Impact of his Ideas About Computability*, eds. G. Sommaruga and T. Strahm, 3–28. Cham: Birkhäuser.

Detlefsen, M. 2005. Formalism. In *The Oxford Handbook of Philosophy of Mathematics and Logic*, ed. S. Shapiro, 236–317. New York: Oxford University Press.

Diamond, C. 1996. Wittgenstein, Mathematics, and Ethics: Resisting the Attractions of Realism. In *The Cambridge Companion to Wittgenstein*, eds. H. Sluga and D. Stern, 226–260. New York/Cambridge, MA: Cambridge University Press.

[103] **Abbreviations of Wittgenstein's Works**

AWL	Wittgenstein (1979)
BB	Wittgenstein (1965)
LFM	Wittgenstein (1976/1989)
PR	Wittgenstein (1980a)
OC	Wittgenstein (1974)
PI	Wittgenstein (2009)
RFM	Wittgenstein (1978)
RPP I	Wittgenstein (1980b)

Dreben, B., and J. Floyd. 1991. Tautology: How Not to Use a Word. *Synthese* 87 (1): 23–50.
Eddington, A.S. 1929. *The Nature of the Physical World*. Cambridge, MA: Cambridge University Press.
Engelmann, M.L. 2013a. Wittgenstein's 'Most Fruitful Ideas' and Sraffa. *Philosophical Investigations* 36 (2): 155–178.
———. 2013b. *Wittgenstein's Philosophical Development: Phenomenology, Grammar, Method, and the Anthropological View*. Basingstoke: Palgrave MacMillan.
Floyd, J. 2001a. Number and Ascriptions of Number in the *Tractatus*. In *Future Pasts: Perspectives on the Analytic Tradition in Twentieth Century Philosophy*, eds. J. Floyd and S. Shieh, 145–192. New York: Oxford University Press.
———. 2001b. Prose versus Proof: Wittgenstein on Gödel, Tarski and Truth. *Philosophia Mathematica* 3 (9): 280–307.
———. 2012a. Wittgenstein, Carnap, and Turing: Contrasting Notions of Analysis. In *Carnap's Ideal of Explication and Naturalism*, ed. P. Wagner, 34–46. Basingstoke: Palgrave MacMillan.
———. 2012b. Wittgenstein's Diagonal Argument: A Variation on Cantor and Turing. In *Epistemology versus Ontology, Logic, Epistemology*, eds. P. Dybjer, S. Lindström, E. Palmgren, and G. Sundholm, 25–44. Dordrecht: Springer Science + Business Media.
———. 2013. Turing, Wittgenstein and Types: Philosophical Aspects of Turing's 'The Reform of Mathematical Notation' (1944 –5). In eds. S.B. Cooper and J. van Leeuwen, 250–253.
———. 2015. Depth and Clarity: Critical Study, Felix Mühlhölzer. In *Braucht die Mathematik eine Grundlegung? Eine Kommentar des Teils III von Wittgensteins Bemerkungen Über die Grundlagen der Mathematik* (Vittorio Klostermann, Frankfurt am Main 2010). *Philosophia Mathematica* 23, Special Issue on Mathematical Depth (2): 255–277.
Floyd, J., and A. Kanamori. 2016. Gödel vis-à-vis Russell: Logic and Set Theory to Philosophy. In *Kurt Gödel: Philosopher-scientist*, eds. G. Crocco and E.-M. Engelen, 243–326. Aix-en-Provence: Presses Universitaires de Provence.
Floyd, J., and F. Mühlhölzer. *Wittgenstein's Annotations to Hardy's a Course of Pure Mathematics: An Investigation of Wittgenstein's Non-Extensionalist Understanding of the Real Numbers*. New York: Springer Verlag, expected date of publication 2018.
Floyd, J. and H. Putnam 2000. A Note on Wittgenstein's Notorious Paragraph About the Godel Theorem. *Journal of Philosophy* 97(11): 624–632.
——— 2012. Wittgenstein's 'Notorious' Paragraph About the Gödel Theorem: Recent Discussions. In *Philosophy in an Age of Science: Physics, Mathematics, and Skepticism*, eds. H. Putnam, M. De Caro, and D. Macarthur, 458–481. Cambridge, MA, Harvard University Press.
Gandy, R. O. 1988. The Confluence of Ideas in 1936. In ed. Herken, 55–112.
Gibson, A. 2010. Francis Skinner's Original Wittgenstein *Brown book* Manuscript. In *Language and World, Part One. Essays on the Philosophy of Wittgenstein. Volume 14 of the Publications of the Austrian Ludwig Wittgenstein Society New Series*, eds. K. Puhl Munz, K. Puhl, and J. Wang, 351–366. Vienna: Ontos Verlag.
Gödel, K. 1931. On Formally Undecidable Propositions of *Principia Mathematica* and Related Systems I. In Gödel (1986), 145–195.
———. 1934. On Undecidable Propositions of Formal Mathematical Systems. In Gödel (1986), 346–372.
———. 1946. Remarks Before the Princeton Bicentennial Conference on Problems in Mathematics. In Gödel (1990), 150–154.
———. 1972. Some Remarks on the Undecidability Results. In Gödel (1990), 305–306.
———. 1986. In *Kurt Gödel Collected Works Vol. I: Publications 1929–1936*, eds. S. Feferman et al. New York: Oxford University Press.
———. 1990. In *Kurt Gödel Collected Works Vol. II Publications 1938–1974*, eds. S. Feferman et al. New York: Oxford University Press.
Goodstein, R.L. 1945. Function Theory in an Axiom-Free Equation Calculus. *Proceedings of the London Mathematical Society* 48: 401–434.

Grattan-Guinness, I. 2012a. Logic, Topology and Physics: Points of Contact Between Bertrand Russell and Max Newman, *Russell: The Journal of Bertrand Russell Studies* n.s. 32: 5–29.

———. 2012b. Manifestations of Logical Pluralism in the Context of Hierarchies, 1900s–1930s. In *The Road to Universal Logic: Festschrift for the 50th Birthday of Jean-Yves Béziau, Vol. II*, eds. A. Koslow and A. Buchsbaum, 93–103. Cham: Birkhäuser/Springer Science + Business Media.

———. 2013. The Mentor of Alan Turing: Max Newman (1897–1984) as a Logician. *The Mathematical Intelligencer* 35 (3): 55–63.

Hardy, G.H. 1929. Mathematical Proof. *Mind* 38 (149): 1–25.

———. 1940. *A Mathematician's Apology*. Cambridge, MA: Cambridge University Press.

———. 1941. *A Course of Pure Mathematics*. 8th ed. Cambridge, MA: Cambridge University Press. 7th ed. 1938. 6th ed. 1933.

Herken, R. 1988 (ed.). *The Universal Turing Machine: A Half-Century Survey*. New York: Oxford University Press.

Hodges, A. 1983. *Alan Turing: The Enigma of Intelligence*. New York/Princeton: Touchstone/Princeton University Press.

———. 1999. *Turing*. New York: Routledge.

Kanamori, A. 2012a. Set Theory from Cantor to Cohen. In *Handbook of the History of Logic, Vol. 6: Sets and Extensions in the Twentieth Century*, eds. D. Gabbay, A. Kanamori, and J. Woods, 1–72. Oxford/New York/Amsterdam: North-Holland/Elsevier.

———. 2012b. The Mathematical Infinite as a Matter of Method. *Annals of the Japanese Association for the Philosophy of Science* 20: 1–13.

Kennedy, J. 2013. On Formalism Freeness: Implementing Gödel's 1946 Princeton Bicentennial Lecture. *The Bulletin of Symbolic Logic* 19 (3): 351–393.

———. 2014. Gödel's 1946 Princeton Bicentennial Lecture: An Appreciation. In *Interpreting Gödel*, ed. J. Kennedy, 109–130. Cambridge/New York: Cambridge University Press.

Klagge, J.C. and A. Nordmann (eds.) 1993. *Ludwig Wittgenstein: Philosophical Occasions, 1912–1951*. Indianapolis: Hackett Publishing Co.

——— (eds.) 2003. *Ludwig Wittgenstein: Public and Private Occasions*. Lanham: Rowman & Littlefield Publishers.

Kleene, S. 1981. Origins of Recursive Function Theory. *Annals of the History of Computing* 3 (1): 52–67.

Kripke, S. 2013. The Church-Turing 'Thesis' as a Special Corollary of Gödel's Completeness Theorem. In eds. Copeland, Posy and Shagrir, 77–104 Cambridge, MA: MIT Press.

Kurz, H.D., and N. Salvadori. 2001. Sraffa and the Mathematicians: Frank Ramsey and Alister Watson. In *Piero Sraffa's Political Economy: A Centenary Estimate*, eds. T. Cozzi and R. Marchionatti, 254–284. London/New York: Routledge.

Lewis, C.I. 1918. *A Survey of Symbolic Logic*. Berkeley: University of California Press.

Lewis, C.I., and C.H. Langford. 1932. *Symbolic Logic*. New York/London: The Century Company.

Littlewood, J.E. 1926. *The Elements of the Theory of Real Functions, Being Notes of Lectures Delivered in the University of Cambridge, 1925*. Cambridge, MA: Heffer & Sons.

Marion, M. and M. Okada. unpublished. *Wittgenstein and Goodstein on the Equation Calculus and the Uniqueness Rule*.

Mays, W. 1967. Recollections of Wittgenstein. In *Ludwig Wittgenstein: The Man and his Philosophy*, ed. K.T. Fann, 79–88. New York: Dell.

McGuinness, B. 1988. *Wittgenstein: A life: Young Ludwig 1889–1921*. Berkeley: University of California Press.

———. 2008. *Wittgenstein in Cambridge: Letters and Documents, 1911–1951*. Malden/Oxford: Blackwell.

Misak, C. 2016. *Cambridge Pragmatism: From Peirce and James to Ramsey and Wittgenstein*. New York: Oxford University Press.

Monk, R. 1990. *Ludwig Wittgenstein: The Duty of Genius*. New York/London: Free Press/Jonathan Cape.

Mühlhölzer, F. 2006. 'A Mathematical Proof Must be Surveyable': What Wittgenstein Meant by This and What it Implies. *Grazer Philosophische Studien* 71: 57–86.

Munz, V.A., and B. Ritter, (eds.) forthcoming. *Wittgenstein's Whewell's Court Lectures, Cambridge, 1938–1941: From the Notes of Yorick Smythies*. Oxford: Wiley-Blackwell.

Newman, M.H.A. 1955. Alan Mathison Turing. 1912–1954. *Biographical Memoirs of Fellows of the Royal Society* 1: 253–263.

Newman, W. 2010. Max Newman – Mathematician, Codebreaker and Computer Pioneer. In *Colossus: The Secrets of Bletchley Park's Code-Breaking Computers*, ed. B.J. Copeland, 176–188. New York: Oxford University Press.

Post, E. L. 1936. Finite Combinatory Processes, Formulation 1. *Journal of Symbolic Logic* 1: 103–105. Reprinted in Post (1994), 103–105.

———. 1941/1994. Absolutely Unsolvable Problems and Relatively Undecidable Propositions–Account of an Anticipation. Reprinted in Post (1994), 375–441.

———. 1944. Recursively Enumerable Sets of Positive Integers and Their Decision Problems. *Bulletin of the American Mathematical Society* 50: 284–316. Reprinted in Post (1994), 461–494.

Post, E.L. 1994. In *Solvability, Provability, Definability: The Collected Works of Emil L. Post*, edited by Martin Davis. Boston/Basel/Berlin: Birkhäuser.

Putnam, H. 1988. *Representation and Reality*. Cambridge, MA: MIT Press.

———. 2015. Naturalism, Realism and Normativity. *Journal of the American Philosophical Association* 1 (2): 312–328.

Quine, W.V. 1940. *Mathematical Logic*. Cambridge, MA: Harvard University Press.

Ramsey, F.P. 1926. The Foundations of Mathematics. *Proceedings of the London Mathematical Society* 2(25): 338–384. Reprinted in Ramsey (1931), 1–61; Ramsey (1990), 164–224.

———. 1927. Facts and Propositions. *Aristotelian Society Supplementary Volume* 7: 153–170. Reprinted in Ramsey (1931), 138–155, Ramsey (1990), 34–51.

———. 1929. General Propositions and Causality. In Ramsey (1990), 145–163.

———. 1930. On a Problem of Formal Logic. *Proceedings of the London Mathematical Society* 2–3 (1): 264–286.

———. 1931. *The Foundations of Mathematics and Other Logical Essays*, edited by R. Braithwaite. London: Routledge & Kegan Paul.

———. 1990. *Philosophical Papers*. Edited by D.H. Mellor. Cambridge, MA: Cambridge University Press.

Russell, B. 1919. *Introduction to Mathematical Philosophy*. London/New York: Routledge. References are to page numbers of the 2nd edition of 1920, published by George Allen and Unwin.

Sen, A. 2003. Sraffa, Wittgenstein, and Gramsci. *Journal of Economic Literature* XLI (December): 1240–1255.

Sieg, W. 2006. Gödel on Computability. *Philosophia Mathematica* 14 (2): 189–207.

———. 2009. On Computability. In *Handbook of the Philosophy of Science: Philosophy of Mathematics*, ed. A. Irvine, 535–630. Amsterdam: Elsevier.

———. 2013a. Gödel's Philosophical Challenge (To Turing). In eds. Copeland, Posy and Shagrir, 183–202.

———. 2013b. *Hilbert's Programs and Beyond*. Oxford/New York: Oxford University Press.

Skinner, F., and L. Wittgenstein. 2015. *The Francis Skinner Archives, Notes and Dictations of Wittgenstein*, Presently Housed at the Wren Library, Trinity College Cambridge.

Solovay, R. 1988. Turing in the Land of O(z). In ed. Herken, 113–148.

Sommaruga, G., and T. Strahm (eds). 2015. *Turing's Ideas – Their Significance and Impact*. Basel: Birkhäuser/Springer.

Sraffa, P. 1960. *Production of Commodities by Means of Commodities: Prelude to a Critique of Economic Theory*. Cambridge, MA: Cambridge University Press.

Stein, H. 1988. *Logos*, *logic*, and *logistiké*: Some Philosophical Remarks on Nineteenth-Century Transformation of Mathematics. In *History and Philosophy of Modern Mathematics*, eds. W. Aspray and P. Kitcher, 238–259. Minneapolis: University of Minnesota Press.

Stern, D., B. Rogers and G. Citron (eds.). forthcoming. *Wittgenstein: Lectures, Cambridge 1930–1933, From the Notes of G.E. Moore*. Cambridge, MA: Cambridge University Press.

Tarski, A. 1933. The Concept of Truth in Formalized Languages. In A. Tarski, *Logic, Semantics, Metamathematics. Papers from 1923 to 1938*, ed. J. Corcoran, 152–278. Indianapolis: Hackett.

Turing, A.M. 1936/1937. On Computable Numbers, with an Application to the Decision Problem. *Proceedings of the London Mathematical Society* 2(43): 544–546.

———. 1939. Systems of Logic Based on Ordinals. *Proceedings of the London Mathematical Society* 2(45): 161–228. Based on Princeton PhD Thesis, 1938. Reprinted in Turing (2001), pp. 81–148 and (with excerpts from Turing's 1938 correspondence) in Copeland (2004), pp. 125–204.

———. 1944/1945. The Reform of Mathematical Notation and Phraseology. Unpublished paper published in Turing 2001, 211–222 and with commentary in Cooper and van Leeuven eds. 2013, 245–249.

———. 1950. Computing Machinery and Intelligence. *Mind* 59 (October): 433–460.

———. 1954. Solvable and Unsolvable Problems. In *Science News 31*, edited by A.W. Haslett, 7–23. London: Penguin Books.

———. 2001. In *The Collected Works of A.M. Turing: Mathematical Logic*, ed. R.O. Gandy and C.E.M. Yates. London/Amsterdam: North-Holland.

Urquhart, A. 2012. Henry M. Sheffer and Notational Relativity. *History and Philosophy of Logic* 33: 33–47.

von Plato, J. 2014. Gödel, Gentzen, Goodstein: The Magic Sound of a G-String. *The Mathematical Intelligencer* 36 (4): 22–27.

Wang, H. 1996. *A Logical Journey: From Gödel to Philosophy*. Cambridge, MA: MIT Press.

Watson, A.G.D. 1938. Mathematics and Its Foundations. *Mind* 47 (188): 440–451.

Webb, J. C. 1990. Remark 3, Introductory Note to Gödel (1972) in Gödel (1990), 281–304.

Weiss, M. 2017. Logic in the *Tractatus* I: Definability. *Review of Symbolic Logic,* 1–50, published online 12 January 2017.

Wisdom, J. 1934. Review of C.I. Lewis and C.H. Langford, *Symbolic Logic*. *Mind* 43 (169): 99–109.

Wittgenstein, L. 1922. *Tractatus Logico-Philosophicus*. London/New York: Routledge & Kegan Paul.

———. 1965. *Preliminary Studies for the "Philosophical Investigations": Generally Known as the Blue and Brown Books*. New York: Harper and Row.

———. 1974. *On Certainty/Über Gewissheit*. Edited by G.E.M. Anscombe and G.H. von Wright, trans. by Denis Paul and G.E.M. Anscombe. Oxford: Basil Blackwell.

———. 1976/1989. *Wittgenstein's Lectures on the Foundations of Mathematics: Cambridge, 1939, from the Notes of R.G. Bosanquet, Norman Malcolm, Rush Rhees and Yorick Smythies*. Edited by C. Diamond. Hassocks: The Harvester Press, Ltd. Reprint, University of Chicago Press.

———. 1978. *Remarks on the Foundations of Mathematics*. Cambridge, MA: MIT Press.

———. 1979. *Wittgenstein's Lectures, Cambridge, 1932–1935: From the Notes of Alice Ambrose and Margaret Macdonald*. Chicago: University of Chicago Press.

———. 1980a. In *Philosophical Remarks*, ed. R. Rhees. Chicago: University of Chicago Press.

———. 1980b. *Remarks on the Philosophy of Psychology. Vol 1.* Edited and trans. By G. H. v. Wright, H. Nyman, C. G. Luckhardt and M. A. E. Aue. Chicago: University of Chicago Press.

———. 1999. *The Published Works of Ludwig Wittgenstein, The Bergen Electronic Edition*. Intelex Corporation, Oxford University Press.

———. 2001. *Ludwig Wittgenstein Philosophische Untersuchungen Kritisch-genetische Edition*, eds. J. Schulte, H. Nyman, E. von Savigny, and G.H. von Wright. Frankfurt am Main: Suhrkamp.

———. 2005. *Big Typescript, TS 213*. Edited and translated by G. Luckhardt and M. Aue. Malden. Blackwell Publishing.

———. 2009. Philosophische Untersuchungen = Philosophical Investigations. Trans. and ed. G.E.M. Anscombe, P.M.S. Hacker and J. Schulte. Chichester/Malden: Wiley-Blackwell.

Wolfram, S. 2013. "Computation, Mathematical Notation and Linguistics", Forward to Turing (1954). In *Alan Turing: His Work and Impact*, eds. B.S. Cooper and J. van Leeuwen, 239–244. Amsterdam/Boston: Elsevier.

Wright, P. 1987. *Spycatcher: The Candid Autobiography of a Senior Intelligence Officer*. New York: Penguin Viking.

Part II
The Universal Machine: From Music to Morphogenesis

Chapter 6
Universality Is Ubiquitous

Martin Davis

Abstract Turing's concept of "universal machine" will be discussed as an abstraction, as embodied in physical devices, as present in nature, and in connection with the artificial intelligence project.

6.1 Introduction

In his deservedly famous article of 1936/37 (Turing 1936/37), Alan Turing changed forever our understanding of what computation is. His analysis based on reducing to its essentials what a person carrying out a computation actually does brought into focus the concept of algorithm in full generality. And his notion of *universal machine* provided a model for today's powerful all-purpose computers. As I've written elsewhere (Davis 2000/2011):

> People had been thinking about calculating machines for a long time, since Leibniz's time and even earlier. Before Turing the general supposition was that in dealing with such machines the three categories, machine, program, and data, were entirely separate entities. The machine was a physical object; today we would call it hardware. The program was the plan for doing a computation, perhaps embodied in punched cards or connections of cables in a plugboard. Finally, the data was the numerical input. Turing's universal machine showed that the distinctness of these three categories is an illusion. A Turing machine is initially envisioned as a machine with mechanical parts, *hardware*. But its code on the tape of the universal machine functions as a *program*, detailing the instructions to the universal machine needed for the appropriate computation to be carried out. Finally, the universal machine in its step-by-step actions sees the digits of a machine code as just more *data* to be worked on. This fluidity among these three concepts is fundamental to contemporary computer practice. A *program* written in a modern programming language is *data* to the interpreter or compiler that manipulates it so that its instructions can actually be executed. In fact Turing's universal machine can itself be regarded as an interpreter, since it functions by *interpreting* successive quintuples in order to perform the tasks they specify.

M. Davis (✉)
Courant Institute, New York University, New York, NY, USA
e-mail: martin@eipye.com

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_6

6.2 Practical Universality

Turing's universal machine is a mathematical abstraction. Unlike material devices with their limitations of space and time, there is no limit to the amount of data that Turing's infinite tape can hold, and no limitation is placed on the number of steps needed to carry out any particular computational process. At best a physical computer can be a practical and useful *approximation* to Turing's universal machine.

Turing's universal machine is constrained by the very weak limited operations that it can carry out, just a local change of a single symbol, and a possible move to a neighboring square. But it makes up for this weakness by its unlimited memory. So the crucial thing for a physical implementation must be a very ample memory. The basic operations in terms of which algorithms will be expressed can be chosen with the principal intended applications in mind. What is important is rapid execution and as large a random access memory as current technology will allow.

In an address to the London Mathematical Society in 1947 (Turing 2004), Turing was quite explicit about the crucial role of a very large memory, approximating the infinite memory of his abstract devices:

> I have spent a considerable time in this lecture on [the] question of memory, because I believe that the provision of proper storage is the key to the problem of the digital computer, and certainly if they are to be persuaded to show any genuine intelligence much larger capacities than are yet available must be provided. *In my opinion this problem of making a large memory available at reasonably short notice is much more important than that of doing operations such as multiplication at high speed [emphasis added].* Speed is necessary if the machine is ... to be commercially valuable, but a large storage is necessary if it is to be capable of anything more than rather trivial operations. The storage requirement is therefore the more fundamental requirement.

In this same lecture, Turing made very clear how he saw the relation between his universal machine of 1936/37, a mathematical abstraction, and his proposal for an Automatic Computing Engine (ACE):

> I considered a type of machine which had a central mechanism, and an infinite memory which was contained on an infinite tape ... Machines such as the ACE may be regarded as practical versions of this same type of machine. There is at least a very close analogy.
>
> Let us now return to the analogy of the theoretical computing machines with an infinite tape. It can be shown that a single special machine of that type can be made to do the work of all. It could in fact be made to work as a model of any other machine. The special machine may be called the universal machine.

Von Neumann, who was proposing his own design in his EDVAC report, was more oblique in his reference to this relationship, preferring to emphasize the importance of pragmatic considerations (von Neumann 1963).

> It is easy to see by formal-logical methods that there exist codes that are in abstracto adequate to control and cause the execution of any sequence of operations The really decisive considerations from the present point of view, in selecting a code, are of a more practical nature: simplicity of the equipment demanded by the code, and the clarity of its application to the actually important problems together with the speed of its handling those problems.

> It would take us much too far afield to discuss these questions at all generally or from first principles.

To see what a profound paradigm shift the new point of view represented, it will suffice to note that as late as 1956, Howard Aiken, a pioneer in the use of electromechanical devices for computation, could write as follows:[1].

> If it should turn out that the basic logics of a machine designed for the numerical solution of differential equations coincide with the logics of a machine intended to make bills for a department store, I would regard this as the most amazing coincidence I have ever encountered.

It is interesting to compare the two proposals: Turing's ACE and von Neumann's EDVAC. Von Neumann's primary interest in numerical calculation was reflected in the logical organization of the EDVAC and its successors:

- ACE was to be used for many tasks for which heavy arithmetic was inappropriate (e.g., chess).
- ACE was organized in a more minimal way with arithmetic operations to be carried out by software rather than hardware.
- The ACE design provided a special mechanism using a stack for incorporating previously programmed operations in a longer program.

Turing's design left a lot to the software, even arithmetic. From the beginning he envisioned a micro-programmable machine with a minimal order code. He scoffed at "the Americans" (Davis 2000/2011):

> [It] is ... very contrary to the line of development here, and much more in the American tradition of solving one's difficulties by means of much equipment rather than by thought ... . Further- more certain operations which we regard as more fundamental than addition and multiplication have been omitted.

6.3 Anachronistic Universality

The very notion that modern computers are intended to approximate universality only began to be accepted in the 1980s. But now that this is generally accepted, there have been a number of strange suggestions that universality should be conferred on certain devices considered or even built before Turing's work.

The *Analytical Engine*, an entirely mechanical programmable calculating device, was proposed by Charles Babbage, but never built. Robin Gandy, a great logician, studied Babbage's work and took the trouble to point out, that if augmented with an infinite memory, the Analytical Engine would be universal (Gandy 1988). But the infinite memory is the whole point, and, not surprisingly, Babbage never realized the importance of a huge random access memory. Of course the basic instruction set suffices. As the Turing machine shows, not much is needed.

[1] Quoted in (Ceruzzi 1983).

Unlike the Analytical Engine, the Colossus was actually built. Its purpose was the decrypting of a class of top-level Nazi command communication, the British called *Fish*. Amazing for its time, it had thousands of vacuum tubes (British: valves). Speaking at the 2009 conference on Unconventional Computing, Benjamin Wells reported that, equipped with an infinite memory, the Colossus would be universal. Again, no surprise.

Arthur & Alice Burks call the ENIAC "general purpose" (Burks and Burks 1981) (which I take to be a synonym for "universal"). Certainly not in the sense Turing had in mind.

More recently Edgar Daylight (Daylight 2012) announced that Howard Aiken's Mark I was a universal machine, even while quoting Aiken's statement a dozen years after Mark I was operational essentially denying the very possibility of such a device.

6.4 Artificial Intelligence

The idea that a computer might have human-like intelligence is based on what one might call *The AI Hypothesis*:

1. The human mind is a function of the brain.
2. The brain is a computer.

If you believe this, the AI project is reasonable. If you reject the possibility of human level AI, you should reject 1 or 2.

Roger Penrose believes 1 but not 2. He argues that 2 implies that our mathematical capabilities would be limited by Gödel incompleteness, whereas we can see that the Gödel undecidable statement is true. Logicians have pointed out to him that what we can "see" is at best that if a given formalism is consistent then the Gödel statement is true. But Penrose remains unconvinced, and expects that deep physics will eventually be shown to provide our brains with a capability that goes beyond what a computer can do (Davis 1990, 1993; Penrose 1989, 1990).

Kurt Gödel, on the other hand held that 2 is correct, while denying 1. He expressed his Cartesian mind-body dualism vividly and poignantly in a consoling letter to the dying Abraham Robinson (Gödel 1974): "The assertion that our ego consists of protein molecules seems to me one of the most ridiculous ever made."

The philosopher John Searle, while praising some AI research, doesn't seem to have expressed an opinion regarding this hypothesis. Instead he uses parables to convince us that whatever wonders computers may exhibit for us, we will not be justified in calling it intelligence. With respect to the defeat of a chess grand master by the computer program Deep Blue, Searle wrote (Searle 1999):

> Here is what is going on inside Deep Blue. The computer has a bunch of meaningless symbols that the programmers use to represent the positions of the pieces on the board. It has a bunch of equally meaningless symbols that the programmers use to represent options for

> possible moves. The computer does not know that the symbols represent chess pieces and chess moves, because it does not know anything.
>
> Imagine that a man who does not know how to play chess is locked inside a room, and there he is given a set of, to him, meaningless symbols. Unknown to him, these represent positions on a chessboard. He looks up in a book what he is supposed to do, and he passes back more meaningless symbols. We can suppose that if the rule book, i.e., the program, is skillfully written, he will win chess games. People outside the room will say, "This man understands chess, and in fact he is a good chess player because he wins." They will be totally mistaken. The man understands nothing of chess, he is just a computer. And the point of the parable is this: if the man does not understand chess on the basis of running the chess-playing program, neither does any other computer solely on that basis.

I commented on this elsewhere (Davis 2000/2011) as follows:

> ... it may not be entirely pointless to bring the Chess Room down to earth. I once had a colleague who had been part of the team that designed Deep Thought, the powerful chess-playing computer that was the predecessor of Deep Blue. He provided me with some numbers on the basis of which I calculated that if the hardware and software constituting Deep Thought were put in the form of a book (more likely a library) of instructions that a human being could carry out, then it would take several years to do the processing needed to make one move. Better put a family in that Chess Room, so the children can take over when the parents die! Otherwise, no game will actually be completed.

Meanwhile, vindicating Turing's prophetic vision, we are the beneficiaries of more and more incredible computer-based devices as memories get larger and larger. Despite Searle and his parables, no one seems to hesitate to speak of a "smart" phone; one doesn't hear: "well of course the phone isn't really smart in the way people are". One can predict that if and when computers have attained the ability to do all of the things that are thought to constitute intelligence in people, people will not hesitate to use the word "intelligence" in referring to that behavior.

6.5 Universality in Nature

Stephen Wolfram has famously argued for the significance of the existence of tiny abstract structures that already exhibit universality (Wolfram 2002). It remains to be seen whether his belief that this aspect of universality will lead to significant scientific discoveries is justified. In particular he has argued that universality is important in biological evolution. I conjecture that he is correct about this, though perhaps not in the way he intends. The long stretches of DNA that lie between genes that had at first been dismissed as "junk" are now more cautiously referred to as "non-coding" referring to the fact that unlike genes, they do not represent a specific protein in coded form. It has been found that considerable portions of this material are conserved by Darwinian evolution over immense stretches of time. My conjecture is that DNA participates in the execution of algorithms that act on germ-plasm and so effect, and I'm guessing, speed up, evolution. At this point, this must be taken as pure speculation coming from someone with not the least expertise. However, here is an example of the kind of thing I imagine: Let us suppose that the DNA includes

a record of past evolutionary development. Next suppose that as part of the reproductive process algorithms modify the germ-plasm based on this record, for example in the direction of increasing certain parameters. If the change is beneficial to survival, Darwinian natural selection will preserve it. If anything like this is correct, then evolutionary change will not have been entirely dependent on random mutations, but will also include a quasi-teleological aspect.

References

Burks, A.W., and A.R. Burks. 1981. The ENIAC: First General-purpose Electronic Computer. *Annals of the History of Computing* 2: 310–399.

Ceruzzi, P.E. 1983. *Reckoners, The Prehistory of the Digital Computer, from Relays to the Stored Program Concept*, 1933–1945. Westport, CT: Greenwood Press.

Copeland, J. (ed.). 2004. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life*. Oxford: Clarendon Press..

Davis, M. 1965/2004. *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. New York: Raven Press/Dover Publications.

———. 1990. Is Mathematical Insight Algorithmic? *Behavioral and Brain Sciences* 13: 659–660.

———. 1993. How Subtle is Gödel's Theorem? More on Roger Penrose. *Behavioral and Brain Sciences* 16: 611–612.

———. 2000/2011. *The Universal Computer: The Road from Leibniz to Turing*. New York: Norton, AK Peters, CRC, Taylor & Francis.

Daylight, E.G. 2012. *The Dawn of Software Engineering: From Turing to Dijkstra*. Heverlee: Lonely Scholar.

Gandy, R.O. 1988. The Confluence of Ideas in 1936. In *The Universal Turing Machine: A Half-Century Survey*, ed. R. Herken, 55–112. New York: Oxford University Press.

Gödel, K. 1974/2003. Letter to Abraham Robinson. In *Kurt Gödel Collected Works Volume V: Correspondence H-Z*, eds. S. Feferman et al., 204–205. New York: Oxford University Press.

Herken, R. (ed.). 1988. The Universal Turing Machine: A Half-Century Survey. In *Oxford*. New York: Oxford University Press.

Penrose, R. 1989. *The Emperor's New Mind*. New York: Oxford University Press.

———. 1990. The nonalgorithmic mind. *Behavioral and Brain Sciences* 13: 692–705.

Petzold, C. 2008. *The Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine*. Indianapolis: Wiley.

Searle, J.R. 1999. I Married a Computer. *The New York Review of Books*, 34–38. April 8, 1999. City.

Turing, A. M. 1936. On Computable Numbers, with an Application to the Decision Problem. *Proceedings of the London Mathematical Society* 2(42): 230–265. Reprinted in Davis (1965/2004) and (with commentary) in Ptzold 2008.

Turing, A. M. 1947/1992. Lecture to the London Mathematical Society 20 February 1947. In *Collected Works of A.M. Turing: Mechanical Intelligence*. ed. D. C. Ince, 87–106 Amsterdam: North-Holland. Reprinted in Copeland (ed.) (2004).

von Neumann, J. 1963. *John von Neumann Collected Works*. New York: Pergamon Press/ Macmillan.

Wolfram, S. 2002. *A New Kind of Science*. Champaign: Wolfram Media.

Chapter 7
The Early History of Voice Encryption

Craig Bauer

Abstract The evolution of voice encryption (a.k.a. ciphony) is detailed from the 1920s through the work of Alan Turing at the end of World War II. The systems covered are invertors, the A-3 Scrambler, SIGSALY, and Delilah. The human cost of poor voice encryption is illustrated by the failure of and consequent lack of trust in the A-3 Scrambler prior to and during World War II. The simple manner in which logarithms and modular arithmetic played a key role in SIGSALY is described, along with other key features. These may be put to use in the classroom to demonstrate practical and extremely important applications of basic mathematical ideas. Alan Turing's work on SIGSALY is detailed along with his efforts to create an even better system, Delilah. Turing's mission to Germany to ascertain the depth of the Nazi's knowledge of ciphony is also discussed. Full historic context is provided to demonstrate the tremendous impact this technology had on the twentieth century.

7.1 Invertors – Better Than Nothing

Ciphony is the science of voice encryption. The earliest voice encryption systems were better than nothing, but not by much. Their story begins in the 1920s when AT&T offered an analog system that simply inverted the speech. That is, low tones were swapped with high tones and vice-versa. We can mathematize this process as follows. First let s denote the frequency of the carrier wave. Then for each component frequency p of the original voice, we convert it to the frequency $s - p$.

The equation reveals a major weakness with this form of encryption. Namely, tones near one half of s are barely changed. So, if you've ever had a professor whose lecturing voice matched Fig. 7.1, and his favorite tone was close to $s/2$, you'd likely be able to understand his lectures even if they were inverted!

Actually, nobody could speak securely using an inverter. Because the system lacked a key, anyone who could build an invertor, which wasn't hard to do, could then decipher the voice as easily as the intended recipient. With practice one can

C. Bauer (✉)
York College of Pennsylvania, York, PA, USA
e-mail: cryptoauthor@gmail.com

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_7

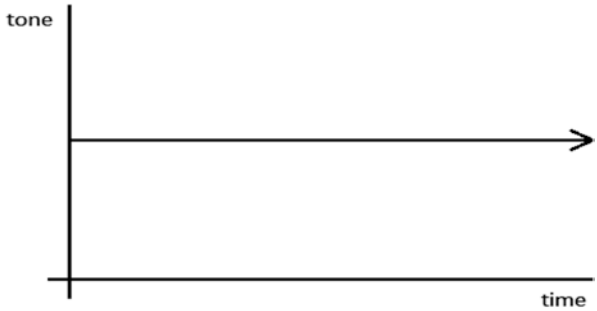


Fig. 7.1 Tone as a function of time for some professors

even understand inverted speech, so the device may not even be needed! In summary, invertors only protect one's privacy against the laziest adversaries.

7.2 The A-3 Scrambler – Protection Against Nosy Neighbors

An improved system was offered by AT&T and RCA in 1937. Called the A-3 scrambler, it split speech into five subbands or channels and then, after inverting some or all of them, shuffled the subbands. Hence, we have a combination of substitution (the inversion of subbands) and transposition (the shuffling). Today's block ciphers make use of both substitution and transposition to attain the confusion and diffusion that Claude Shannon argued should be present in a strong system, but the technique predates Shannon's work. Indeed, Fritz Nebel designed systems (ADFGX and ADFGVX) with these features and they were put to use by the Germans in World War I. Although this was a step in the right direction for both text and voice, neither Nebel's systems nor the A-3 scrambler proved secure. For the A-3, the problem was that the ***keyspace*** was too small. With only five subbands, there are a mere 5! = 120 ways to reorder them. Since we have a choice of inverting or not inverting each of the subbands, there are 32 possibilities altogether in that regard, and the total keyspace is (120)(32) = 3840. This is small enough that a brute-force attack (simply trying all of the possibilities) will succeed. Yet, the situation is even worse than this indicates. Of the 3840 keys, most were unsuitable for use, as they left much of the speech understandable. It was finally decided that only 11 keys were safe to use, and of these, only 6 were actually chosen. They were implemented in a cycle with 36 steps, each of which had a duration of 20 s. Hence, after 12 min, the key cycle would repeat (Kahn 1996, p. 554).

In order to understand the level of protection afforded by the A-3 Scrambler, it is useful to make an analogy. It is desirable for the front door of a home to offer some security, but inside the home we settle for much weaker locks on bedroom and bathroom doors. These are merely intended to offer privacy, not security. Someone attempting to enter an occupied bathroom in a home will be unable to open the door, as usual, and will then simply walk away. The lock is not meant to keep out a determined adversary. Similarly, a scrambler offers some protection against nosy

neighbors for someone on a party line,[1] but cannot be expected to hold against the determined attacks of rival nations' intelligence organizations.

In strong contrast to the typical nosy neighbor, Nazi Germany had some of the best mathematicians and engineers in the world and tremendous resources that could be applied to breaking voice encryption systems of other nations. Yet President Franklin D. Roosevelt and Prime Minister Winston Churchill communicated with each other using the A-3 Scrambler! It only took the Germans a few months to break it and they were listening to the deciphered results by September 1941 (Kahn 1996, pp. 555–556).

Neither the Americans, nor the British, were as naïve as their use of this insecure system might indicate. They both knew they should've been using something better.

> The security device has not yet been invented which is of any protection whatever against the skilled engineers who are employed by the enemy to record every word of every conversation made. (British foreign Office Memorandum, June 1942).[2]

> In addition, this equipment furnishes a very low degree of security, and we know definitely that the enemy can break the system with almost no effort. (Colonel Frank McCarthy, Secretary to the Army General Staff, October 1943).[3]

Since the quotes above show that the Americans and the British knew the system they were using for voice encryption offered no security, we have to answer the obvious question of why they persisted in using it.

The short answer is that it was the best system they could devise at the time. The longer answer (and the reason behind the short one) is that speech is very difficult to encrypt, compared to text. The main reason for this is the incredibly degree of redundancy in human speech. We can usually accurately reconstruct text following the deletion of the vowels. Some written languages don't even bother to write the vowels. We can even reconstruct text with as much as 50% of it removed in many cases. This is because the redundancy of English is around 50%. With spoken English having an even higher degree of redundancy, severe mangling still allows recovery. We can think about this in another way. An ideal ciphertext should look random. The extreme non-randomness of the source (in the case of speech) makes the conversion more challenging. There is also the technical hurdle of dealing with sound, as opposed to a small set of symbols as in alphanumeric text.

---

[1] Younger readers will likely require an explanation of the term "party line." As a first step, imagine a house with phones that actually connect to jacks in the walls (i.e. land lines). A boy upstairs might pick up the phone in his room and hear his dad talking to someone. He'd realize his dad was using the downstairs phone and hang up. All of the phones in the house would go out via a common line. This would be convenient for conference calls, but inconvenient the rest of the time. A family member would sometimes have to wait his turn, when wanting to make a call. "Party lines" worked on the same principle, but the phones were in different homes. That is, in the old days, you might be on a party line with one or more neighbors. You could listen in on their calls, if you desired, but would hopefully respect their privacy and hang up when you discovered the line was in use.

[2] FO/371/32346. Taken here from (Hodges 1983), p. 236.

[3] From a letter to to Harry Hopkins, Assistant to President Roosevelt. Taken here from (Mehl 1997), p. 5.

A communication made using the A-3 scrambler can be attacked with a sound spectrograph. This graph of the enciphered speech's tones may be cut up along visible breaklines and then reassembled to look smooth. The process is much like completing a jig-saw puzzle.

Making the jump from the A-3 Scrambler's 5 channels to 10 or 20 channels only offers a marginal improvement. It certainly increases the keyspace, but the "jig-saw puzzle attack" offered by the sound spectrograph will still work. To truly secure speech, a new scheme was needed, not just an upgrade.

7.3 The Cost of Insecurity

But why is the problem of ciphony so important? Can't any spoken message be sent as text instead? An historic example shows why this is not an acceptable solution.

American cryptanalysts succeeded in breaking Japan's diplomatic ciphers prior to the attack on Pearl Harbor. The difficult machine cipher that lay behind the encipherments was codenamed Purple and one of the decrypts revealed that Japan was about to break off diplomatic relations with the United States. While this cannot be immediately equated with an imminent attack on Pearl Harbor, it did indicate war. Due to the geographic location of Pearl Harbor, it was clear that a warning should be issued, so that preparations could be made in case of attack. A phone call would obviously be the quickest way to issue the alert, but General Marshall didn't trust the A-3 scrambler. A Japanese eavesdropper on his conversation would likely realize that Marshall couldn't be aware of Japan's intent to break off relations unless the diplomatic cipher had been compromised. Following this realization, the cipher system would be replaced and America would miss out on all of the intelligence the intercepts could convey until her cryptanalysts could make another break – a process that could take many months, or perhaps prove impossible. Fearing this possibility, and not knowing an attack really was imminent, General Marshall conveyed his warning via text. The delay caused by this resulted in the warning being received after the attack.

If America had a voice encryption system in place that could be trusted, the quicker verbal warning would have certainly saved many lives, as well as ship tonnage. These in turn would have allowed a quicker more powerful response to Japan's aggression, and therefore a quicker end to the war, once more saving lives.

7.4 A Solution from the Past

There are far fewer technical hurdles to contend with for encrypting text. We saw earlier in this paper how the combination of substitution and transposition was attained for text during World War I, but not for speech until 1937. In a similar manner, an old technique for encrypting text, the one-time pad, would now be applied to

voice encryption.[4] When used properly, text enciphered in this manner was theoretically unbreakable, but could it provide comparable security for voice? Before answering that question, a summary of how the one-time pad works for text is provided. It is easily demonstrated through an example.

Given a message such as "I LOVE ALICE," a one-time pad of random numbers (as long as the message) may be used to encipher it. Suppose the numbers are 20, 18, 13, 7, 16, 11, 5, 8, 24, 20. Shifting each letter in the message by the number indicated in the same position of the pad gives C DBCU LQQAY. Notice that when the shift amount takes us past Z, we continued on from the start of the alphabet. Also, since the text we are shifting consists of letters from an alphabet of size 26, we shift by random values between 0 and 25, inclusive. If the person who intercepts the message can guess the random numbers that were used to encipher, he can shift each of the ciphertext letters back and recover the original message. But the interceptor has no reason to favor one random number string over another. If he guesses the key as 20, 22, 1, 9, 16, 11, 5, 8, 24, 20, the message will turn out to be "I HATE ALICE." On the other hand, the key 20, 18, 13, 7, 16, 19, 22, 24, 0, 11 yields the message "I LOVE SUSAN." In fact, any possible ten letter message can be obtained if the corresponding key is tried. This is what makes the one-time pad unbreakable. The would-be decipherer has no way of telling which message was intended, for he has no reason to favor one random key over another.

As the name indicates, a one-time pad should not be used twice. Reuse of a particular key will allow both messages enciphered by it to be recovered by any competent cryptanalyst. America's cryptanalysts were able to break reused Soviet one-time pads even when the keys were used years apart by different individuals. Other examples of one-time pad exploitations exist.[5]

Nevertheless, used properly, the one-time pad is unbreakable and the idea may be transferred from text to voice. One simply adds random values to the sound wave to encipher and then subtracts them out again to decipher. This technique has almost nothing in common with the inversion and reordering of subbands used previously. It is simple in theory, but presented large engineering obstacles to put into practice. This is the challenge that was to be solved by SIGSALY.

7.5 SIGSALY

The Following are Equivalent:

1. SIGSALY
2. RC-220-T-1
3. The Green Hornet
4. Project X-61753

[4] This system was published in 1882, but ignored or forgotten, and finally reinvented in 1917–1918. The original discovery was only recently recognized. For details see Bellovin (2011).

[5] See Bury (2009, 2011).

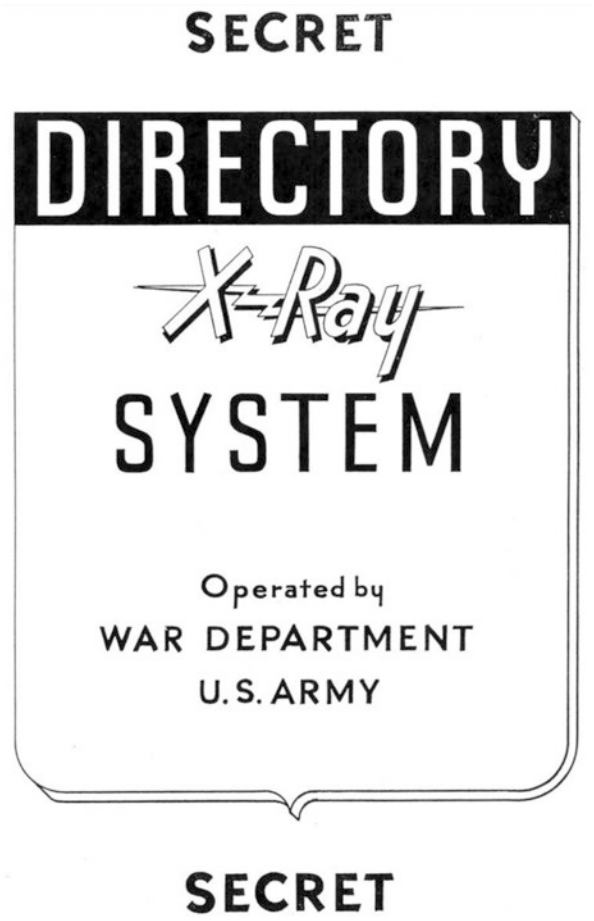


Fig. 7.2 Is this how we should market texts?

5. Project X (The Atomic Bomb was Project Y.)
6. X-Ray
7. Special Customer

Proof – see the literature.

As indicated above, SIGSALY, the ciphony system that would replace the A-3 scrambler for Roosevelt and Churchill (and others), had many different names. This is an indication of its importance.

The sixth name may be seen on the cover of a formerly classified directory for the system (Fig. 7.2). The cover is certainly attention grabbing, but the contents are quite dry, by comparison.

Before getting into the details of how SIGSALY worked, a few pictures are presented (Figs. 7.3 and 7.4).

Fig. 7.3 A view of SIGSALY (Image from http://www.cryptologicfoundation.org/content/A-Museum-Like-No-Other/COMSEC.shtml)

If you're examining the images above looking for the particular piece of equipment (SIGSALY) under discussion, you're missing the forest for the trees. Today's voice encryption devices can fit in your palm, but the images above only show portions of SIGSALY. The engineering challenge posed by transferring the one-time pad to speech resulted in a 55 ton behemoth that occupied 2500 square feet. Today, areas of this magnitude more commonly describe homes than electronic equipment. Indeed electronics is not a field where bigger equals better! So why was SIGSALY so large?

Necessity is the mother of invention, so it's not surprising that a new system was devised following the failure of the A-3 scrambler and the disaster at Pearl Harbor. But the actions of Japan and the Nazi threat meant that the new system had to found quickly. Its lack was costing lives. Something secure had to be created and what it looked like or how much space it took up didn't matter. The only important constraint was time. Thus, any existing technology that could be incorporated was; it saved time. A voice coder (*vocoder* for short) served as the basis for the new system. Vocoders digitized speech for the purpose of data compression. They promised some commercial appeal, since compressed messages would travel through undersea phone cables using less bandwidth, and therefore reducing costs for businesses. The result of this compression was analogous to lowering the resolution of an image. It might not sound like the person you're talking to, but you could still under-

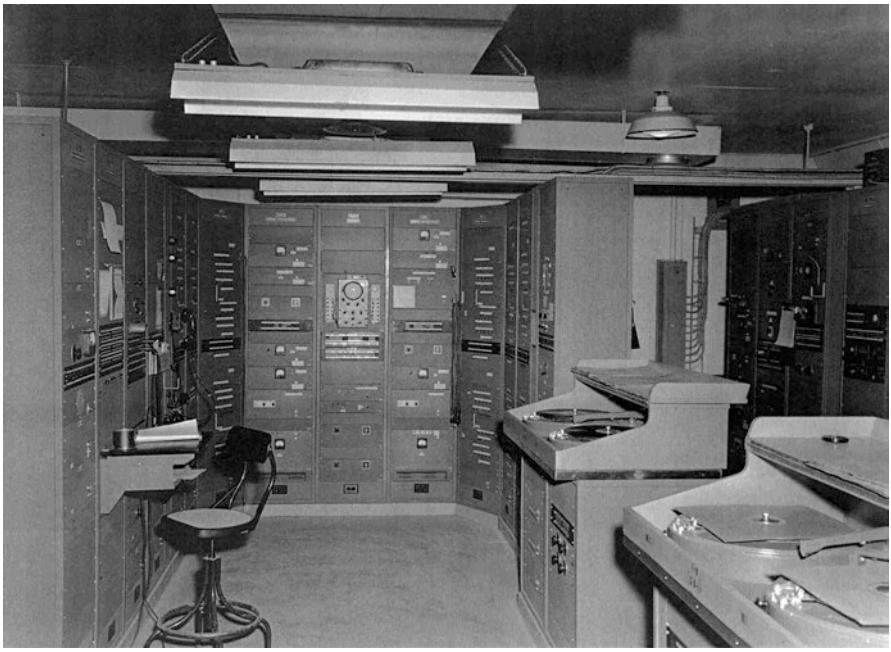


Fig. 7.4 Another view of SIGSALY (From the National Archives and Records Administration)

stand the words. Earlier in this paper I avoided stating a figure for the redundancy of speech, but the vocoders proved it was at least 90% by demonstrating that a voice message compressed to 10% of its original bandwidth could still be understood.[6]

Nothing in the theoretical formulation of SIGSALY required compression, but it was a nice feature to have nevertheless. The important thing was to digitize the speech. Once this was done, a random digital key could be combined with the speech to simulate the one-time pad. The available vocoders were large, so even before the encryption began, SIGSALY was big and heavy.

Frank Zappa remarked "Writing about music is like dancing about architecture." Since the sound of a vocoder is best conveyed by listening to one, I direct you to a recording of a Bell Labs vocoder from 1936 at http://www.complex.com/music/2010/08/the-50-greatest-vocoder-songs/bell-telephone-laboratory.

You've likely heard vocoder speech before, even if the term is new to you. For example, the voice of the Cylons in the 1970s TV series Battlestar Galactica was produced by a vocoder.[7]

Long before this show aired, Secretary of War Henry Stimson described the sound of a vocoder:

> It made a curious kind of robot voice.[8]

[6] (Tompkins 2010), p. 23.

[7] A cylon from a 1977 episode of Battlestar Galactica may be heard at http://www.youtube.com/watch?v=0ccKPSVQcFk&feature=endscreen&NR=1

[8] (Tompkins 2010), p. 63.

Stimson is famous for closing down America's Cipher Bureau in 1929, during his tenure as Secretary of State, and remarking later, "Gentlemen do not read each other's mail." By World War II, he changed his mind concerning the need for cryptanalysis.

All of this brings us to an interesting point. Vocoders sound cool. For this reason, many musicians have made use of them. Dave Tompkins, a hip-hop journalist, aware of the use of vocoders in voice encryption and music, wrote a very entertaining book that examines both applications. The front cover of this book appears below (Fig. 7.5).

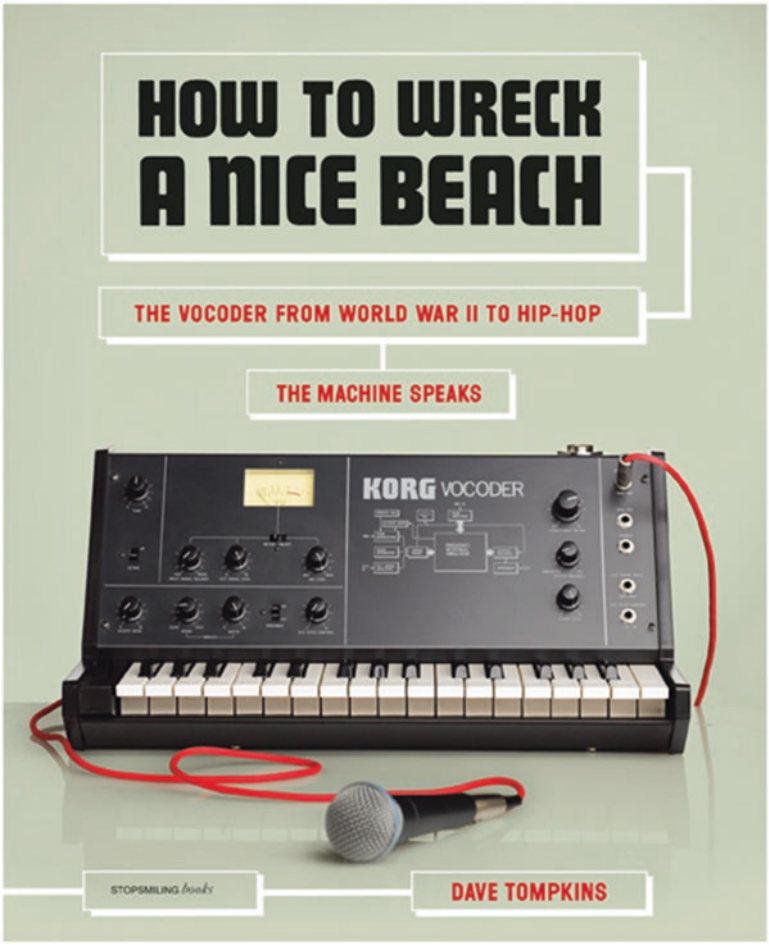


Fig. 7.5 For a book with cryptologic content, Tompkins's work contains a record-shattering amount of profanity

Fig. 7.6 Decades later, these men knew nothing about the use of vocoders in music (Mehl 1997), p. 45

The title of Tompkins book arose from the manner in which vocoders were tested. Various phrases would be passed through the vocoders and listeners, ignorant of what they were supposed to hear, would try to determine the messages. In one instance, the phrase "How to recognize speech" was misheard as "How to wreck a nice beach." Clearly that vocoder was not suitable to military applications in which a slight misunderstanding could have a calamitous effect.

The diverse applications of the vocoder, detailed in Tompkins's, book are represented by Figs. 7.6 and 7.7.

SIGSALY's vocoder converted the speech into 10 channels (from 150 Hz to 2950 Hz) and another channel represented pitch. Some sources describe the pitch as being represented by a pair of channels. Both points of view can be considered accurate, as will be made clear in the next paragraph. Since each of SIGSALY's channels took up 25 Hz, the total bandwidth (with two pitch channels) was (12)(25) = 300 Hz. The final transmissions were made at VHF.

We might expect the amplitude of any channel, measured at a particular point in time, to be labeled as either 0 or 1. Indeed, an attempt was made to use this binary scale. However, the result was too low resolution. The speech could no longer be understood.[9] The scale that proved workable was senary: amplitudes were measured

[9] (Hodges 1983), p. 246.

Fig. 7.7 Musicians, represented here by Michael Jonzun (and a Roland SVC vocoder), knew nothing of the use of vocoders by the military (Tompkins 2010)

using values between 0 and 5, inclusive. But the pitch proved to be more sensitive. Ultimately it was decided to represent pitch using a scale that ranged from 0 to 35, inclusive. Since such a scale can be represented by a pair of numbers between 0 and 5, pitch may be regarded as consisting of two channels.

Before we get to modular arithmetic, the mathematical star of this portion of the tale, we examine how logarithms made a contribution to winning the war.

Once it was decided to use a senary scale for the amplitude, it would seem time to move on to determining other aspects of the system. Yet, there remained an opportunity to optimize the manner in which the amplitude was represented. The obvious linear scale, where an amplitude of 2 is twice as large as an amplitude of 1 and an amplitude of 3 is three times as large as 1, etc., is not the best approach. The reason is that humans do not perceive sound in that manner. We can distinguish low amplitude sounds more finely than high amplitude sounds. Therefore, it makes more sense to use a logarithmic scale than a linear scale. The result is that sounds we can distinguish more easily will in fact be distinguished by our scale, while other sounds with the same difference between them (but sounding more similar to us) will be lumped together. Hence, it will be easier for us to understand the compressed vocoder representation of the speech.

Applying a logarithmic scale in this context is referred to as "Logarithmic Companding." The person who coined this term demonstrated his love of compression as he did so, for "companding" is a compression of "compressing/expanding".[10]

The concept of measuring sound amplitudes logarithmically, as described above, should already have been familiar to all readers, whether they realize it or not, for who hasn't heard of the Decibel scale? It's a logarithmic scale and it is how we're used to describing sound.

With each channel of the speech now represented by a string of integers between 0 and 5, we may begin the encryption. The random key for each channel is also a string of integers ranging from 0 to 5, but we do not add them in the normal manner. The result of the standard addition, if greater than or equal to 6, must be reduced by 6. That is, the arithmetic is performed modulo 6. Hence, the encrypted channel will consist solely of numbers between 0 and 5, inclusive.

An example is given in Fig. 7.8 below.

Why was the addition of the key done in this complicated manner? Why not just add without the mod 6 step? Three reasons are given below.

1. The mod 6 step was Harry Nyquist's idea.[11] Students of information theory will recognize this name and, for them, it certainly lends a stamp of authority to support the inclusion of this step. But, an argument from authority is not a proof! Fortunately, we have two more reasons.
2. If the sum is not reduced modulo 6, then an enciphered value of 0 can only arise in one way: a 0 in the signal combining with a 0 in the key. Similarly, an enciphered value of 10 guarantees that both signal and key took the value 5. A nontrivial portion of the message is thus given away to any interceptor who knows the system.

   To put it numerically, $2/36 \approx 5.5\%$ of the signal could be quickly determined without any doubt, by any interceptor, as described above, if the modulo 6 step is omitted.
3. Without the modulo 6 reduction, the key would have an effect on the speech similar to that of random background noise. Are you able to understand a friend talking despite the sound of an air-conditioner or chainsaw running in the background?

A key feature of SIGSALY's encryption process was that there was a separate and independent random key for each channel. It wasn't safe to use the same key for all of the channels! A simplified sketch of the encryption and transmission process is provided by Fig. 7.9. The voice channels are represented by spectrum 1 through spectrum 10. There is also a rectangle labeled "pitch detector" to represent the pitch channel (or pair of channels, if you prefer). As the diagram indicates, this takes place prior to making use of the key. Other steps that I do not describe here must be made, but these "missing steps" are of greater interest to engineers than mathematicians. The curious reader may turn to Donald E. Mehl's book.

[10] The pitch channel, however, wasn't companded.

[11] (Mehl 1997), p. 38.

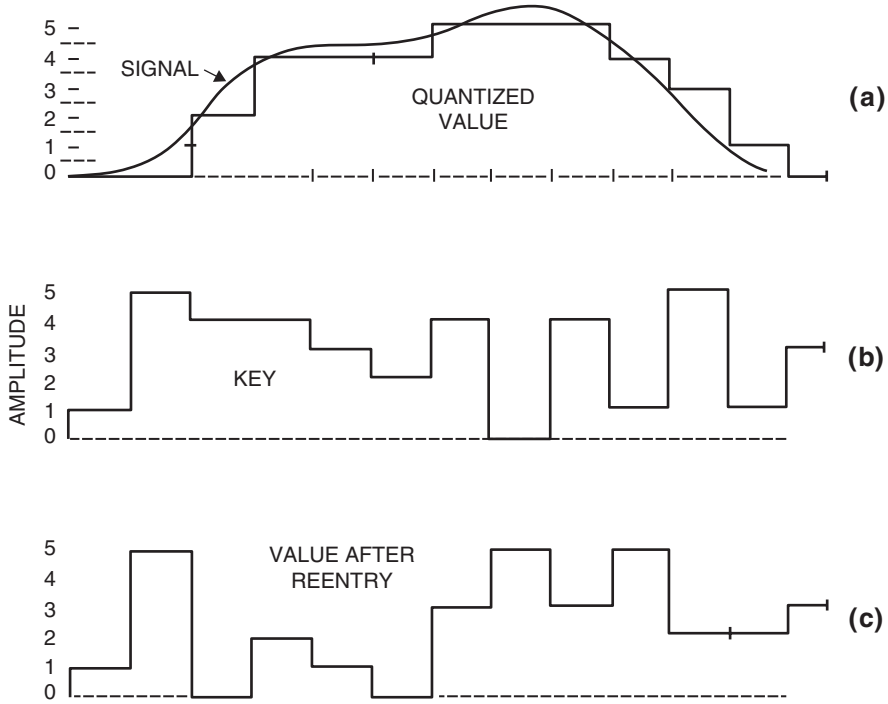


Fig. 7.8 The mod 6 addition of the key was referred to as "reentry" by the creators of SIGSALY (Image from (Mehl 1997), p. 40, and (Boone and Peterson 2000), p. 19)

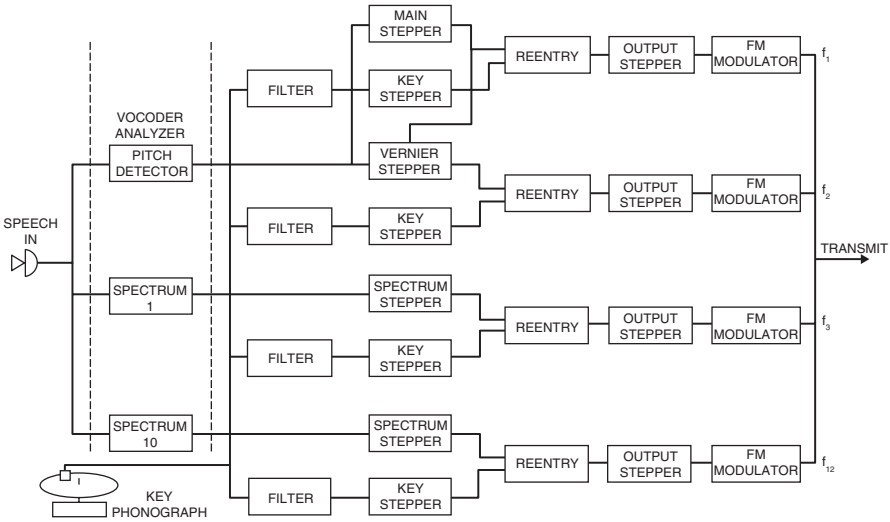


Fig. 7.9 A simplified sketch of SIGSALY's encryption and transmission process (Mehl 1997, p. 26)

Fig. 7.10 A SIGSALY turntable and record, with a modern CD for scale (Mehl 1997, p. 31)

At this point I'd like to draw your attention to the lower left hand corner of Fig. 7.9. The "key phonograph" is exactly what it sounds and looks like. The source of the keys that needed to be combined with each channel was simply a record (Fig. 7.10).

The one-time key for voice encryption was codenamed SIGGRUV. As with text, the key was added to encipher and subtracted to decipher. Taking the form of a 16 inch record, it was larger than the vinyl used on home turntables. In normal operation, each record could provide 12 min worth of key. However, if the record were to stop playing for some reason, communication would be automatically cut off. This prevented an unknowing user from continuing to broadcast without the key, and therefore without encryption. On the receiving end, an identical record, playing in synchronicity, would allow the same key to be subtracted out again, modulo 6, to recover the original vocoder voice.

Since it is the "grooves" of a record that convey sound through the phonographs needle and onto the speakers it would be natural to assume that the codename suffix GRUV was chosen for that reason. But codenames, for extra security, should never be chosen to correspond in any way to the secrets they protect. In this case, the apparent meaning was simply a coincidence.

The vocoder's discrete approximation of speech had a sampling rate of 50 times per second. In other words an estimate of the amplitude of the user's voice was made every fiftieth of a second, and each of his 10 voice channels, as well as the two pitch channels, had to be combined with a separate key value at each of these instants. To make this possible, the SIGGRUV records had to simultaneously produce 12 tones, at various frequencies, and change them randomly fifty times per second. However, this is not the whole story. There were also tones that had nothing to do with encryption. A tone at one frequency, for example, served to help keep the enciphering and deciphering records synchronized.

A sampling rate of 50 times per second may seem arbitrary, but there is sound reasoning behind it. Syllables are not the most basic unit of speech. They may be decomposed into phonemes, which last for about a fiftieth of a second each. Setting the sampling rate to coincide with these fundamental units allows them to be treated separately. Just as the key for a one-time pad must change with every letter, so we must also change our voice encryption key with every phoneme to have a true analog to the unbreakable text system. Less frequent sampling would correspond to changing the one-time pad key only after every few letters. On the other hand, sampling the speech more often did not seem necessary and would make the transatlantic synchronization of the records even more difficult.

The fact that the key for each channel should be random deserves more attention. The problem of generating randomness is very difficult and persists today. For example, in 2012 thousands of RSA private keys were recovered through an attack that exploited the not-so-random manner in which the large primes needed to implement the system were generated for the users.

During World War II, the random sequences demanded by SIGSALY were obtained by recording thermal noise. As an added twist, it was not used as is for the key, but reversed first in time. That is, the keys consisted of backwards playing thermal noise.

Each of the records was destined to be a one-hit wonder, for like the one-time pad, reuse could compromise the messages. Thus, over 1500 such key sets were made[12] and after each secure conversation, the serviceman filling the role of SIGSALY dj was to take the request "Don't Play it Again Uncle Sam!" To be sure he complied with this request, each record was destroyed following airplay.

7.6 Plan B

For one time pad systems, whether text or voice, key distribution is a problem. A separate key is needed for every message and since none can be reused, it is not especially desirable to have duplicates lying about. However, a spare in the case of SIGSALY is not a terrible idea. For if only one copy is held at a given location and it gets misplaced, broken, or just scratched, the conversation cannot take place. So, in the early days of SIGSALY, spares were made for key records. When things were seen to be running smoothly, worries decreased and the production of key records dropped to the bare minimum. The operators could also take comfort in the fact that there was a Plan B in the event that the records or even the turntables themselves became inoperable. We now take a look at this backup system.

As was mentioned earlier, size and weight weren't the constraints SIGSALY's designers were laboring under. Codenamed SIGBUSE, SIGSALY's back-up key resembled a locker room with some odd items thrown in for good measure (Fig. 7.11). The purpose of SIGBUSE was simply to generate a pseudorandom key, if there ever arose a reason the records couldn't be used. These back-up keys, gener-

[12] (Tompkins 2010), p. 68.

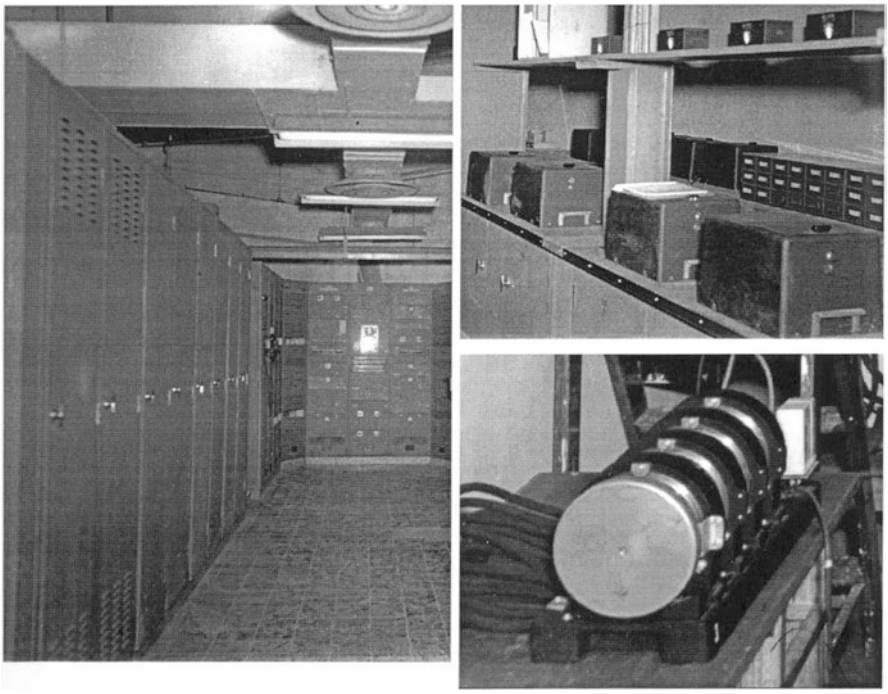


Fig. 7.11 Not the locker room, but rather SIGSALY's back-up key SIGBUSE (Mehl 1997, p. 34)

ated mechanically, were known to be less secure, but it was better than having no back-up and, if they were ever compromised, there would be no effect on the security of the records, once they were back in use.

Both America and Great Britain were aware that the A-3 scrambler wasn't secure. Since SIGSALY was intended to replace this weak device, allowing Roosevelt and Churchill to communicate quickly and securely, it was important for both sides to feel that they could trust it. To complicate matters, its initial design was entirely an American effort and the operating teams, even in the proposed London installation, would be made up of Americans, who would hear everything.

This is where Alan Turing enters the story. As England's greatest cryptanalyst, he was dispatched to America to evaluate SIGSALY. But there was tremendous resistance to allowing *any* foreigner to get a glimpse of this machine! We don't have proof, but it appears that the debate, about whether or not to let Turing in on the work, reached President Roosevelt.[13] In any case, Secretary of War Stimson resolved it in Turing's favor.

On 19 January, 1943, Turing entered Bell Labs and began his study. He spent a great deal of time on cryptanalysis, trying to find weak spots in SIGSALY's design.In the end, it seems that all he had to offer were some improvements to the SIGBUSE

[13] (Mehl 1997), p. 69, (Hodges 1983), p. 245, and (Tompkins 2010), p. 59, all believe the matter reached Roosevelt.

key. In regard to having Americans operate the proposed London installation, he reported, "If the equipment is to be operated solely by U.S. personnel it will be impossible to prevent them listening in if they so desire." However, the Americans were typically concentrating so much on their required tasks for operating the machine that they were not aware of what was being said by those using it. Based on Turing's reports, the British eventually decided that the security SIGSALY offered was acceptable. Roosevelt and Churchill now had a replacement for the A-3 Scrambler.

Turing's examination of SIGSALY inspired him to create his own (completely different) system, Delilah. Turing's report on Delilah appeared publicly for the first time in the October 2012 issue of *Cryptologia* (Turing and Bayley 1946).

It might seem that Turing's work on SIGBUSE was pointless, since the SIGGRUV system never had a failure that necessitated the use of this backup, but the experience he gained played a role in another system he would design after returning to England.

A more critical part of SIGSALY was the air-conditoning system. It is shown in Fig. 7.12. A voice encryption system that fills a house requires a cooling system on the same scale!

Fig. 7.12 SIGSALY's air conditioning system (Mehl 1997, p. 50. Mehl appears on the right in this photo)

7.7 SIGSALY in Action

The first (experimental) SIGSALY installation was located in New York in November 1942, nearly a year after Pearl Harbor. Finalized installations in Washington D.C. and London went into operation in July 1943. This marked not only the first secure conversation between Roosevelt and Churchill, but many other firsts. The vocoder technology described earlier was off-the-shelf in the sense that it was pre-existing technology, but it hadn't actually seen use. Thus, SIGSALY represented the first practical "Pulse Code Modulation" (referring to the digitization process) and the first broadcasting and reception of digital speech.

Figure 7.13 indicates the various SIGSALY installations that followed the first tests. Obviously some installations (such as Hawaii) could be established earlier and more easily than others (such as Berlin and Tokyo).

Figure 7.14 provides yet another view of a SIGSALY installation. In this one, the phone is clearly visible, but this is not what the caller would be using. The phone you see was used by a member of the operating team to make sure synchronization was being maintained.

A separate room existed to allow the user(s) to converse in a more comfortable condition (Fig. 7.15).

Although technology rapidly diminished the space needed for secure voice encryption, JFK's system (Fig. 7.16) looked decidedly less cool. It looked like something Maxwell Smart might have used. What would the next step be, three phones?

7.8 SIGSALY Retires

SIGSALY received an honorable discharge, having never been broken. The Germans never even realized that it represented speech. They conjectured that it was meaningless noise or maybe a teletype signal. SIGSALY transmissions resembled the theme music from the American TV program *The Green Hornet*. It's unlikely that many Nazis knew this show, but the Americans did, and this led to the most informal name for SIGSALY.

Although SIGSALY was never broken, General Douglas MacArthur didn't trust it! Happily, others did, and the rewards of the instant communication it provided were reaped. Given its success, it's natural to ask why it wasn't kept in use longer. There were several reasons:

1. It weighed 55 tons and had a 70 ton shipping weight.
2. It took up 2,500 square ft.
3. It cost between $250,000 – $1,000,000+ per installation.
4. It converted 30 kilowatts of power into 1 milliwatt of low quality speech.[14]
5. The deciphered speech sounded like Donald Duck.[15]

[14] (Hodges 1983/2012), p. 247.

[15] General Eisenhower complained that it made his wife sound like an old woman. The system was optimized for male voices, and as a result, deciphered female voices sounded worse.

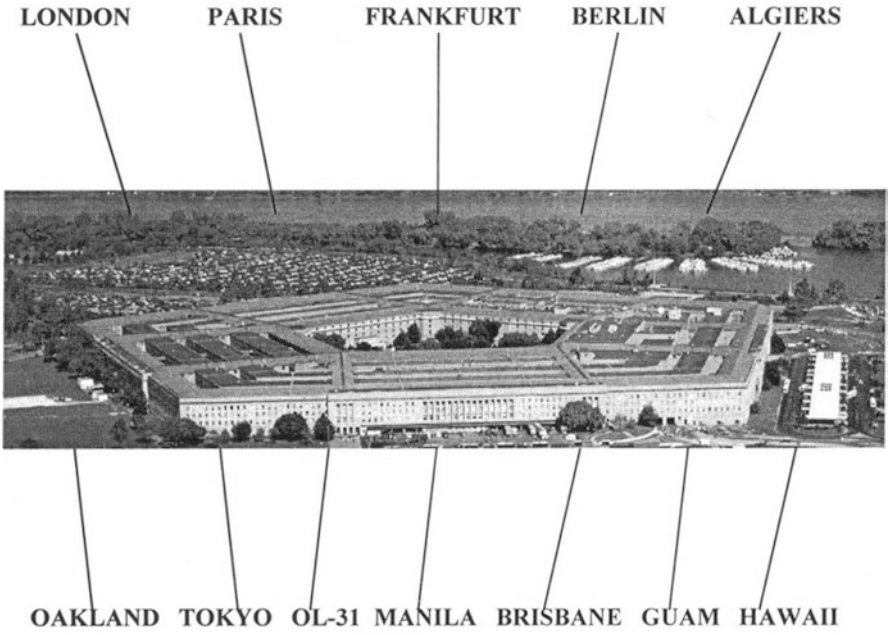


Fig. 7.13 The SIGSALY installations. OL-31 was on a barge (Mehl, 1997, p. 86)

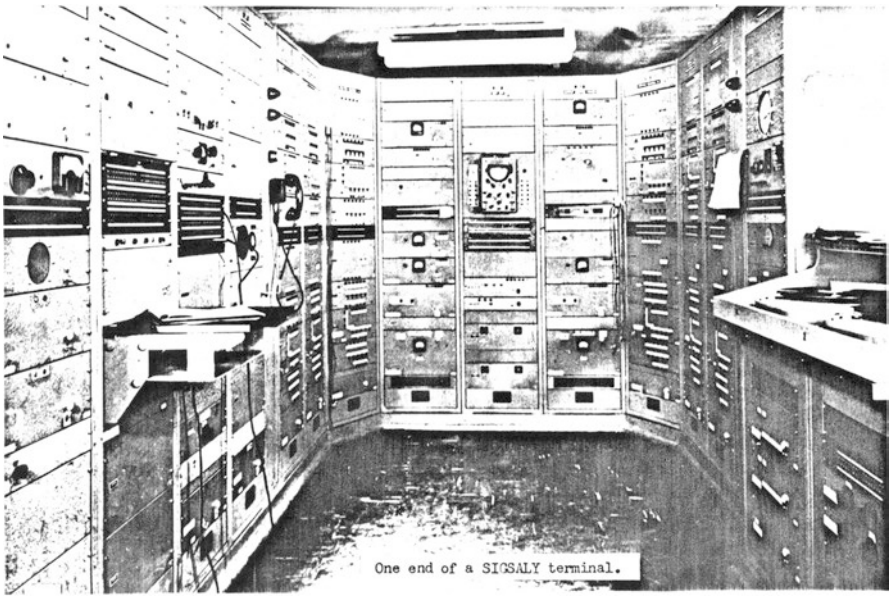


Fig. 7.14 Yet another view of SIGSALY (Image courtesy of The National Cryptologic Museum, David Kahn Collection, VF 12-7)

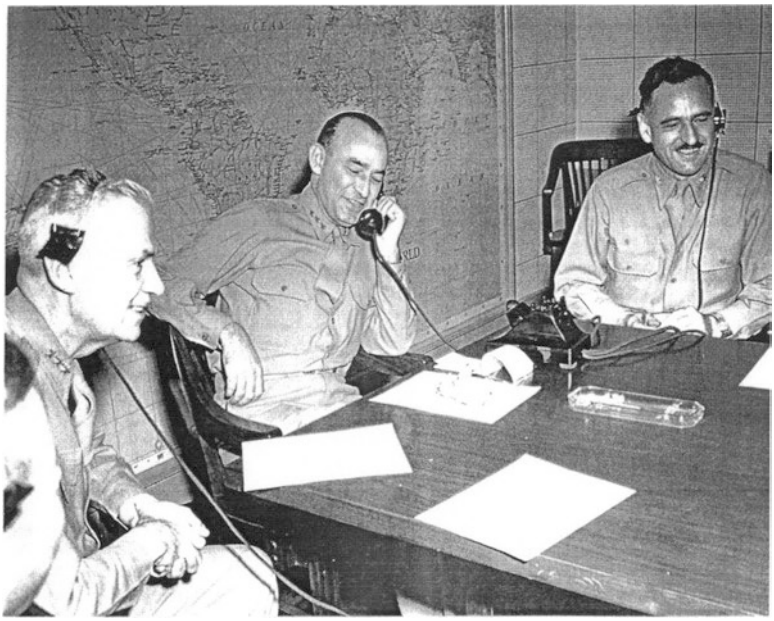


Fig. 7.15 SIGSALY Users – Fighting the Germans and Japanese… and loving it! (Mehl 1997, p. 103). An alternate caption for this image is "SIGSALY: Your Digital Pal Who's Fun to be With!"

Newsday Center Section

Guarding the privacy of communication of President Kennedy, as well as the security of high level diplomatic conversations around the world is the telephone scrambler, which electronically jumbles speech. The device's strange appearance, as shown above in a portable model, is matched only by the unusual role it has played and is playing in world affairs.

THE SOUND OF SECRECY

By David Kahn

radiotelephones in his official car is equipped with a scrambler. The White

broken simply by repeated listening. The so-called New Zealand device, which com-

Fig. 7.16 JFK's voice encryption system

NEW YORK TIMES, PAGE 27

JULY 3, 1976

From WWII era

'Green Hornet' patent awarded

Franklin D. Roosevelt

Winston Churchill

By STACY V. JONES
New York Times Service

WASHINGTON — A historic note was struck in the inventors' world this week when basic patents were granted for a secret method of speech transmission used by President Franklin D. Roosevelt, Prime Minister Winston Churchill and other Allied leaders in World War II.

Bell Telephone Laboratories Inc., as assignee of the inventors, received the two patents, both entitled "secret telephony." They had been applied for almost 35 years ago, but were, until recently, classified secret by the government.

The patents were awarded to Robert C. Nathes and Ralph K. Potter. P. W. Blye was project manager in charge. All three engineers have long since retired from the Bell staff.

THE X System, as it was called, was developed in the early 1940s for the government and was the first speech encoding technique that could not be deciphered by anyone but the intended recipient. The user spoke into a specially designed handset. The speech was encoded with electronic pulses called "key signals" and transmitted by shortwave radio. At the receiving end, the key signals were subtracted and the listener heard only the clear message.

A person receiving the message had to have a phonograph record of the identical key-signal pattern being transmitted. The recordings had to be started simultaneously at both ends, and new pairs were made every day.

The X System, which was nicknamed "The Green Hornet," after a popular radio show of the 1940s, was the precursor of digital speech encoding and a forerunner of the present-day "pulse code modulation" which is used in the Bell System to transmit encoded conversations over telephone channels.

Fig. 7.17 A 1976 New York Times article on SIGSALY (Image courtesy of The National Cryptologic Museum, David Kahn Collection, Folder 12-7)

SIGSALY was finally declassified in 1976. This allowed a slew of patents, applied for decades earlier, to finally be granted (Fig. 7.17).

A mock-up of a portion of SIGSALY may be seen today at the National Cryptologic Museum (Fig. 7.18). This museum also has an excellent library that includes the David Kahn Collection (Hamer 2011). Kahn is widely regarded as cryptology's greatest historian and, prior to his donation, his collection was the largest in private hands.

Early in this paper we saw the consequences that may be faced when a nation is without a secure voice encryption system. The next section takes a look at the advantage gained when one is possessed.

7.9 Voice vs. Text

Text systems take longer to encipher and decipher than voice systems. The situation was far worse during the pre-computer era of World War II. Then, an enciphered message might take an hour to reach readable form. Sometimes this was too long to

Fig. 7.18 The National Cryptologic Museum's SIGSALY mock-up (See http://www.nsa.gov/about/cryptologic_heritage/museum/index.shtml)

wait! The instant communication voice encryption allows can make a tremendous difference when speed is of the essence. The best example of this is provided by another voice system – the Navajo code talkers. The rapid communication made possible by these men allowed for equally rapid and coordinated movement of troops, in response to changing conditions. This was an advantage the Japanese did not possess.

> Were it not for the Navajos, the marines would never have taken Iwo Jima! (Major Howard M. Conner[16])

Like SIGSALY this was a "voice system" that was never broken. But code talkers couldn't be used forever, while digital voice encryption has been continuously improved up to the present day (Fig. 7.19).

[16] Paul (1973), p. 73.

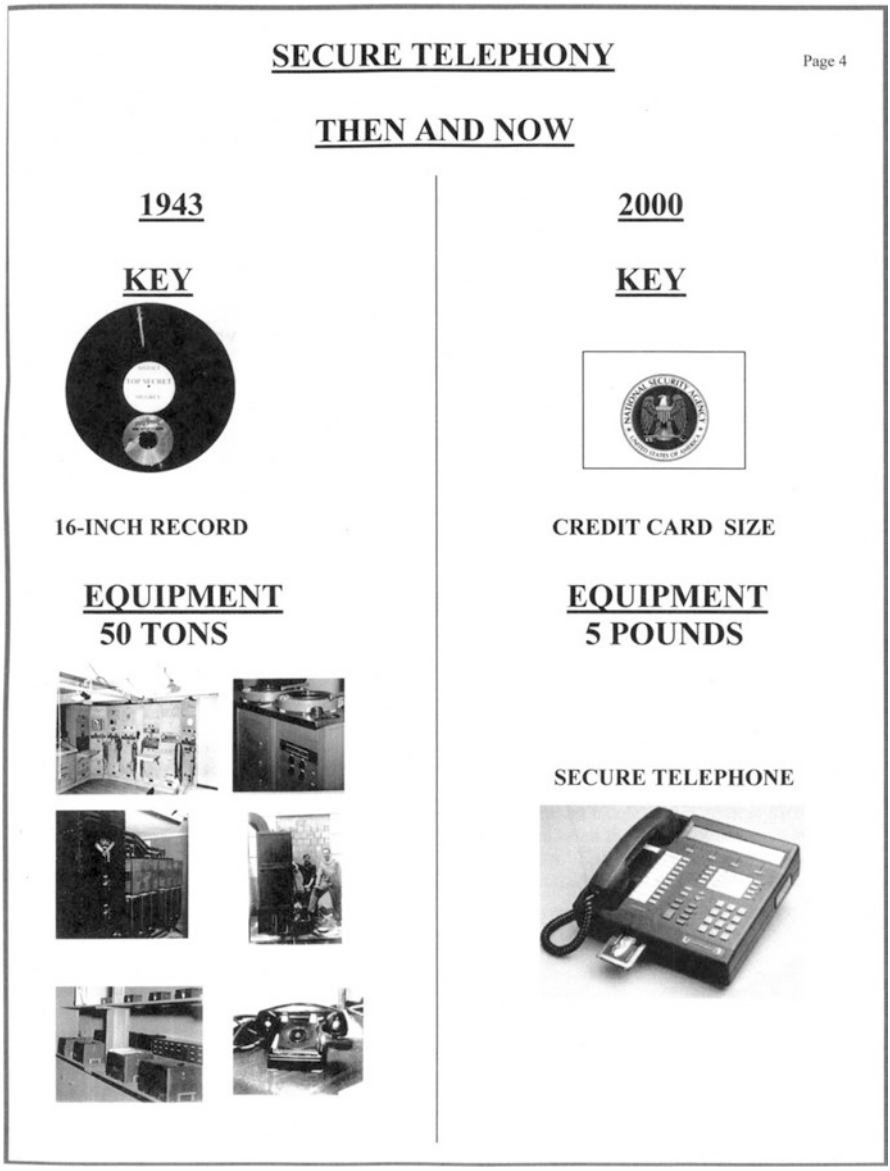


Fig. 7.19 A comparison between SIGSALY and a secure telephone from the year 2000 (Image from Reunion 2000, 805th Signal Service Company, Washington D.C., October 2000, p. 4, courtesy of The National Cryptologic Museum, VF 60-38)

7.10 Turing's Take on the Problem

Although Turing was a very experienced cryptanalyst when he left England for the purpose of evaluating SIGSALY, it is not known that he had any special experience with voice encryption. It seems that his visit provided him with a quick and intense course in this special form of encryption. Of course it's an understatement to say that he was a quick learner. On his way back to England in March 1943, he already had the conception of another, possibly superior, way to encipher speech (Hodges 1983/2012, p. 273).

Turing's work on his proposed voice encryption system was different from his previous work in nearly every way. For starters, he carried out the research at Hanslope Park, 10 miles north of Bletchley Park, where his earlier work was done. He was nearly working alone, instead of as part of a large team. It should be noted though that he did have help from Donald Bayley, an electrical engineer. While Turing's work at Bletchley Park was extremely important for a very large number of people, his work in ciphony at this time appears to have been solely for his own entertainment (Hodges 1983/2012, p. 269–270). With a working system (SISALY) in place, there was absolutely no demand for anything else.

That's not to say that there wasn't room for improvement. Recall the following "biographical information" for SIGSALY:

1. It weighed 55 tons and had a 70 ton shipping weight.
2. It took up 2,500 square ft.
3. It cost between $250,000 and $1,000,000 per installation.
4. It converted 30 kilowatts of power into 1 milliwatt of low quality speech.
5. The deciphered speech sounded like Donald Duck.

With the above faults in mind, Turing decided to pursue a more elegant solution. Despite having an extremely creative mind, he relegated the task of naming his proposed voice encryptor to another by making its naming a contest. Whoever could come up with the best name would receive a small prize. The result was "Delilah." (Hodges 1983/2012, p.273). Since the Biblical Delilah was a "deceiver of men" and a successful encryption also deceives men, the name satisfied Turing, despite his being an atheist.

Although it has been known for decades that Turing created Delilah, significant new information became available in recent years. First, Ralph Erskine found "'Delilah Report on Progress" in the British National Archives (HW 62/6). In this short piece, Turing provided a description of the proposed machine:

> The unit uses seven valves and when suitably rearranged will probably occupy a space of about 10" x 8" x 5".

The Americans did come up with a smaller version of SIGSALY, called "Junior X". It employed an 8 channel vocoder and fit in a van. But this is still significantly larger than Turing's description of Delilah and, for what it's worth, Junior X never saw use in the field (Tompkins 2010, p. 54).

So how did Turing get such a tremendous reduction in size? He basically scrapped everything that wasn't absolutely essential. In particular, he got rid of the vocoder component and attempted to encipher the analog voice signal directly. He also jettisoned the idea of having both a primary key (SIGSALY's records) and a back-up key (SIGBUSE). Instead, Turing made sole use of a keying system similar to SIGBUSE in some ways. It should be recalled that Turing's main contribution to SIGSALY was offering an improvement to the SIGBUSE portion.

By far the biggest revelation concerning Delilah was a technical paper authored by Turing and Bayley that was finally released to the British National Archives in 2009 (Turing and Bayley 1946). By the time I found out about it, it was checked out by the British Government. GCHQ's historian expressed surprise when I informed him of this. He said that it should not have been checked out without his approval. Luckily, I was able to obtain a scan of the paper from someone who saw it before me. Although the scan did not include the diagrams, which would have been useful, I was able to retype the text so that it could be run in the pages of *Cryptologia's* October 2012 issue, in time for the end of the Turing Centennial. It quickly became *Cryptologia's* most read article.

In addition to directly encrypting analog signals, Delilah differed from SIGSALY by sampling the speech 4000 times per second. Turing conversed with Claude Shannon at Bell Labs and seems to have picked this sampling rate based on what's known as "Shannon's Bandwidth Theorem," although it should be attributed to Whittaker (1915). Namely:

> If a signal time function is sampled instantaneously at regular intervals and at a rate at least twice the highest significant signal frequency, the samples contain all of the original message.[17]

Applying this theorem for Delilah's 2000 Hz gives the sampling rate of 4000 samples per second. Decades later, CDs would deliver 44.1 kHz with a 16 bit (65,536 levels) sampling accuracy and DVD Audio is at 192 kHz with a 24 bit (16,777,216 levels) sampling accuracy.[18] But for the time period, 4,000 samples per second was a very impressive rate.

While signal amplitudes for SIGSALY were modeled by integers between 0 and 5, Delilah's were scaled to max out at 1 and the key was combined with the signal modulo 1. Yet Delilah was not using the binary addition that failed to make SIGSALY messages intelligible. "Modulo 1," in the context of an analog system, means that we simply through away the integer portion of the amplitude after performing the addition, and keep the fractional part. Results will be numbers such as .375 and .819. We are not limited to just 0 and 1. Although the mod step behaves differently for Delilah, it is still essential. Without it, the effect would, once again, be like hearing someone over background noise.

Turing's unique approach resulted in some unique difficulties. For instance, the enciphered speech was too high frequency for telephone circuits. Turing's answer to

[17] Taken here from (Mehl 1997), p. 71.

[18] It's incredible to compare these parenthetical values with SIGSALY's 6 levels!

this was to "feed each "spike" into a specially devised electronic circuit with an orthogonal property" (Hodges 1983/2012, p. 275). Called "Turing's Orthogonal Circuit," these devices are still used and typically referred to as "matched filters." The idea is actually due to Nyquist.[19] Today's digital communications systems commonly use this technique to avoid intersymbol interference (ISI). After passing through the filter, the zero crossings will be at the sampling points, and therefore won't cause interference.

Turing and his coauthor Bayley described two possible approaches for providing Delilah with a key (1946, p. 37):

> (a) recording random noise on discs or tape and using those recordings simultaneously at the ends of the transmission path and (b) generating identical voltages at each end.

Clearly option (a) was inspired by SIGSALY. But how could Turing synchronize at 4000 samples per second, when it was challenging at a rate of 50 samples per second for SIGSALY?

He and Bayley admitted that (a)

> …has the disadvantage that the mechanical difficulties of starting and maintaining the keys in synchronism are large, and, furthermore, the number of discs or reels of tape required becomes prohibitive.

Left to generate pseudorandomness somehow, option (b), the men came up with the following idea (Turing and Bayley 1946, p. 37):

> Six multivibrators are locked with the pulse from the combiner and their outputs taken to networks which serve both to isolate individual multivibrators and to differentiate the outputs (so as to strengthen, relatively, the higher harmonics.) The fundamental frequency of each multivibrator is some exact sub-multiple of the combiner pulse frequency, the various sub-multiples being: 5, 7, 8, 9, 23, and 31.

We can see the influence his work on Enigma had, as Turing's description of Delilah's keying mechanism continues:

> The outputs of these networks (26 in all) pass through the cypher machine and are combined at the output end to form seven inputs to seven distorting networks. The distorting networks have differing phase characteristics so that the two out-puts which are produced by combining their seven outputs at the plugboard depend enormously on what frequencies were fed into the various networks i.e. on the setting of the cypher machine and the plugboard (Turing and Bayley 1946, p. 37).

> Note: It has been assumed above that the reader is familiar with the cypher machine used. Should that not be the case it will be sufficient to understand that the machine is a device enabling 26 contacts to be connected to 26 others in a pre-determined random manner and that the mode of connection may be changed by pressing a key (Turing and Bayley 1946, p. 38).

The submultiples selected, 5, 7, 8, 9, 23, and 31, are relatively prime values. Today's stream ciphers combine linear feedback shift registers of relatively prime length to ensure a large period. This is indeed what Turing had in mind, but of

[19] See (Nyquist 1928), p. 632.

course, the key will still be periodic and with a sampling rate of 4000 times per second, it doesn't take long to reach the end of it!

> The multivibrators start simultaneously and since they have fundamental frequencies which are prime to one another it follows that they will arrive back at the starting position after a time 5x7x8x9x23x31 times the period of the locking pulse. That is, the key will repeat after 1,785,600 x 250 secs 7.48 mins (Turing and Bayley 1946, p. 38).

A repeating key would completely compromise this system. It would be no different than replaying a SIGSALY record or reusing a one-time pad. Therefore, when using Delilah, it was necessary to pause and change keys every 7 min. This could be accomplished by changing any of the rotor positions or plugboard connections.

As was indicated before, Turing carried out much of this work by himself, and since there was no pressing need for an improved voice encryption system, he didn't have much in the way of funding or resources. Without the support one would normally expect for such a challenging project, he was unable to reach a satisfactory result. In particular, the synchronization issue was never fully resolved. While "local calls" could be made, no transatlantic conversations were possible. The system could never fill SIGSALY's role.

Another difficulty was measuring and transmitting the amplitudes precisely enough. The simpler means of approximating amplitudes implemented in SIGSALY caused less trouble.

Yet another problem was the 10 dB Signal to Noise ratio. This translates into the speech being only one order of magnitude stronger than the noise. Turing and Bayley admitted that this was "rather lower than desired."

Earlier in this paper I pointed out that the most important constraint on a secure speech system during World War II was time. It had to be ready fast. Delilah simply wasn't completed in time (spring 1945). If there was a commercial market for such equipment, it again failed, since in addition to other problems the recovered speech was of very low quality.

Turing's premier biographer, Andrew Hodges provided the following blunt summary:

> As a contribution to British technology it had been a complete waste of time (1983/2012, p. 346).

But was it a waste of time for Turing, who lived for intellectual challenges? Speaking much more generally, John Lennon pointed out, "Time you enjoy wasting, was not wasted." I have the feeling that Turing enjoyed the time he spent with Delilah.

In a similar way, working from the Bayley/Turing report, John Harper is in the process of rebuilding Delilah. Thus, thanks to John, we may soon be able to hear and evaluate the quality of Delilah ourselves.

Turing's final encounter with voice encryption took place in Germany, and it is this episode that we know the least about.

7.11 Turing in Germany

It had been reported that Turing traveled to Germany very soon after the end of World War II. Personally, I was skeptical of these claims. Why would England risk the life of her top cryptanalyst in such a potentially hazardous mission? Yet, when I put the question to GCHQ's historian, I received confirmation of the story. Turing was, in fact, sent to Feuerstein, Germany. This location lay in Upper Franconia, Bavaria, and housed a vocoder lab. Turing was not the only "big name" to go on such potentially dangerous missions. Tommy Flowers went with Turing on at least one trip.

Being an expert on voice encryption, Turing was asked to determine how far German research had advanced in this direction. The short answer is "not far." Although the Germans had broken the A-3 scrambler, they didn't have anything better to use for their own voice communications.

Even major details of this mission remain ambiguous. Tompkins places Turing's visit to Feuerstein on May 15, 1945, prior to the TICOM[20] raid on it that summer (Tompkins 2010, pp. 190–4). Hodges mentions Turing making a trip to Germany in July, but does not name Feuerstein (1983/2012, pp. 311–312). It is not known if these were two separate trips or if dates somehow got confused and they represent the same mission. It is also possible that there is no error in the above clams. If Turing was part of the TICOM raid in the summer of 1945 (which seems likely), and the date matches what Hodges claims, then everything is consistent.

There may be much more left to tell of such trips. Prior to World War II, Göttingen University was one of the best institutions in the world for mathematics. The Nazis put an end to that, so why did Turing, as Juliet Floyd noted, make trips to Göttingen towards the end of his life? It has been claimed that this helped German science advance.[21]

Although there have been several items relating to Turing declassified in recent years, it is known that more material remains out of public view. What other government work did Turing engage in during his final years?

Acknowledgments The National Security Agency's Center for Cryptologic History made it possible for me to take a close look at ciphony and many other fascinating cryptologic topics by supporting me full-time as their 2011–2012 Scholar-in-Residence. Portions of this paper previously appeared in an NSA in-house publication. The National Cryptologic Museum's librarian, Rene Stein, also played a very important role in helping me locate materials I needed. Wayne Blanding patiently answered basic engineering questions and Dave Tompkins spiced up presentations I give on this topic by sharing his knowledge of vocoders in music through his entertaining book and responses to my emails. John Harper helped me understand Delilah and Tina Hampson delved into the British Archives on my behalf. Thank you all!

[20] Standing for Target Intelligence Committee, this group was tasked with recovery of cryptologic (as well as other) intelligence. Although it was led by the U.S., the British were also part of the effort.

[21] Juliet Floyd called my attention to Lohrengel and Heinz (2010), which states that Turing was in Göttingen in 1948 and 1950, stimulating astrophysicists working on computing to develop new methods of data storage.

References

Bellovin, S.M. 2011. Frank Miller: Inventor of the One-Time Pad. *Cryptologia* 35 (3): 203–222.

Boone, J.V., and R.R. Peterson. 2000. *Sigsaly – The Start of the Digital Revolution.* Center for Cryptologic History, National Security Agency, Fort George G. Meade, Maryland, July 2000. Available online at http://www.nsa.gov/about/cryptologic_heritage/center_crypt_history/publications/sigsaly_start_digital.shtml.

Bury, J. 2009. Breaking Unbreakable Ciphers. The Asen Georgiyev Spy Case. *Cryptologia* 33 (1): 74–88.

———. 2011. From the Archives: Breaking OTP Ciphers. *Cryptologia* 35 (2): 176–188.

Hamer, D. 2011. The David Kahn Collection at NSA's National Cryptologic Museum. *Cryptologia* 35 (2): 110–113.

Hodges, A. 1983/2012. *Alan Turing: The Enigma*. New York/Princeton: Touchstone/Princeton University Press.

Kahn, D. 1996. *The Codebreakers*. 2nd ed. New York: Scribner.

Lohrengel, S., and F. Heinz. 2010. *Eine Entwicklung, die die ganze Welt betrifft*, Göttingener Tageblatt, 22 October 2010.

Mehl, D.E. 1997. *The Green Hornet.* Self-published.

Nyquist, H. 1928. Certain Topics in Telegraph Transmission Theory. Transactions of the AIEE, Vol. 47, pp. 617–644, Apr. 1928, Presented at the Winter Convention of the AIEE, New York, NY, February 13–17, 1928.

Paul, D.A. 1973. *The Navajo Code Talkers*. Pittsburgh: Dorrance Publishing Co., Inc..

Tompkins, D. 2010. *How to Wreck a Nice Beach*. Chicago: Stopsmiling Books.

Turing, A., and D. Bayley. 1946. Government Code and Cypher School: Cryptographic Studies, HW 25/36, *Report on Speech Secrecy System DELILAH,* a Technical Description Compilied by A.M. Turing and Lieutenant D. Bayle REME, 1945–1946, British National Archives, released in 2009. Published 2012 in *Cryptologia* 36(4): 295–340.

Chapter 8
Turing and the History of Computer Music

B. Jack Copeland and Jason Long

Abstract The story of Turing's pioneering work in creating the first computer-generated musical notes in Manchester in 1948–1949 is told, as well as the story of Christopher Strachey's work (later Oxford's first professor of computing), who extended Turing's note-playing routines to create computer-generated melodies. Recordings were made in Turing's Computing Machine Laboratory by the British Broadcasting Corporation (BBC) in 1951: by analyzing Turing's programming manual for the Manchester machine—the first ever written for a stored-program computer—and utilizing retrospective computer analysis of the recordings, a kind of "digital archaeology" is employed in order to reconstruct the Turing-style routines that were used to play the music recorded by the BBC. These techniques have also enabled us to restore the recordings. We establish Turing's leading role in the history of computer music.

8.1 Introduction

One of Turing's contributions to the Digital Age that has largely been overlooked is his pioneering work on transforming the computer into a musical instrument. It's an urban myth of the music world that the first computer-generated musical notes were heard in 1957, at Bell Labs in America.[1] In fact, about nine years earlier, computer-generated notes were heard in Turing's Computing Machine Laboratory at Manchester University.

This chapter analyzes Turing's groundbreaking work at Manchester, and also describes how Christopher Strachey, later Oxford University's first professor of computing, used and extended Turing's note-playing routines to create

[1] See, for example, Chadabe (2001).

B.J. Copeland (✉)
University of Canterbury, Christchurch, New Zealand
e-mail: jack.copeland@canterbury.ac.nz

J. Long
Victoria University of Wellington, Wellington, New Zealand
e-mail: jason.long@ecs.victoria.ac.nz

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_8

computer-generated melodies. Computer-generated music was recorded in Turing's Manchester Laboratory by the British Broadcasting Corporation (BBC) in 1951.[2] We outline the techniques of digital archaeology, and the sheer detective work, which led to our reconstruction of the routines that were used to play the music in this recording, and led also to our restoration of the recording. To set the scene, we begin with an overview of the historic Manchester computer and of the characters involved in our story. Then we turn to Turing's own programming manual for the Manchester machine—the first ever written for a stored-program computer. We clarify his notation for the machine's instructions, and trace in detail how his lines of code were actually transformed into computer-generated notes. We describe the computer-assisted search that helped us to reverse engineer the Turing-type melody-playing algorithms used to produce the music in the recording—algorithms that ushered in the field of computer music. Our analyses establish Turing's pioneering role in the history of computer music.

8.2 The Hardware

The world's first electronic all-purpose stored-program computer—the first electronic universal[3] Turing machine—ran its first program in June 1948 (Copeland 2011a, b). Called simply "Baby", this historic machine was tiny, rough and ready, and almost entirely lacking in the facilities needed for serious computing. Programs were entered bit by painstaking bit, using a panel of hand-operated switches to plant each bit in memory. Output was in the form of bright dots on a tiny glass screen. Baby was designed by two brilliant engineers, Freddie Williams and Tom Kilburn, to test their new groundbreaking high-speed electronic memory, the Williams tube. Around an array of three Williams tubes they wired together the simplest stored-program computer they could think of (Fig. 8.1).

In later life Kilburn was at pains to deny that Turing had contributed anything to the Baby (Copeland 2011b). This was highly misleading. For one thing, Turing contributed the fundamental concept, the very idea of a universal machine that stores programs of symbolically-coded instructions in its memory. Summarizing his pre-war work, Turing wrote in 1947:

> Some years ago I was researching on what might now be described as an investigation of the theoretical possibilities and limitations of digital computing machines. I considered a type of machine which had a central mechanism, and an infinite memory which was contained on an infinite tape. ... [D]igital computing machines ... are in fact practical versions

[2] Part of this recording can be heard at http://www.abc.net.au/classic/content/2014/06/23/4028742.htm. This edition of *Midday with Margaret Throsby* (ABC Radio National, 23 June 2014) is a musical tour through Turing's life and work, on the 102nd anniversary of his birth.

[3] Of course, it was a universal machine with a *finite* memory, a concept introduced by Turing in his "Intelligent Machinery" (1948), where he spoke (p. 422) of "a universal machine with a given storage capacity".

Fig. 8.1 Baby, the first electronic stored-program computer. Baby came to life in June 1948. The proud parents: Tom Kilburn is on the left, Freddie Williams on the right (Courtesy of the University of Manchester School of Computer Science)

> of the universal machine. There is a certain central pool of electronic equipment, and a large memory, [and] the appropriate instructions for the computing process involved are stored in the memory (Turing 1947).

Kilburn had stepped into Turing's world at the end of 1946, when he entered a dingy London lecture room and sat down to listen to Turing explaining how to build a computer.[4] Williams had recently succeeded in storing a single binary digit on the face of a cathode ray tube, proving that his computer memory idea worked in principle; and so, as Williams said, "the point now had been reached where we'd got to find out about computers".[5] They heard that Turing was giving a series of lectures on computer design in London, and it was decided that Kilburn would attend.[6] The lectures ran from December 1946 through to February 1947, and were held in a conference room at the Adelphi Hotel in the Strand.[7] Kilburn was a good pupil, quickly progressing during the lectures from not knowing (as Williams put it) the

[4] The lecture notes are published as "The Turing-Wilkinson Lecture Series (1946-7)" (Turing (1946-7). The series of nine lectures (about half of which were given by Turing's assistant, Jim Wilkinson, most likely from notes prepared by Turing) covered Versions V, VI, and VII of Turing's design for the ACE; see also Copeland (1999).

[5] Williams in interview with Christopher Evans in 1976. ("The Pioneers of Computing: An Oral History of Computing", London: Science Museum. © Board of Trustees of the Science Museum. Transcription by Copeland (1997)).

[6] Bowker and Giordano (1993), p. 19.

[7] See Copeland (2005), pp. 459–464. Womersley's handwritten notes concerning the arrangements for the lectures (Woodger Papers, catalogue reference M15) are in *The Turing Archive for the History of Computing* <http://www.AlanTuring.net/womersley_notes_22nov46>).

"first thing about computers"[8] to the point where he could start designing one himself. Kilburn's initial design (later superseded) for what would eventually be the Manchester computer followed Turing's principles closely, and Kilburn's written reports made extensive use of the terminology Turing had taught him in the Adelphi lectures.[9] When asked where he had got his basic knowledge of the computer from, Kilburn usually said, rather irritably, that he couldn't remember.[10] In a 1993 interview, he commented vaguely "Between early 1945 and early 1947, in that period, somehow or other I knew what a digital computer was", adding "Where I got this knowledge from I've no idea".[11] There is in fact no mystery about where Kilburn got his basic knowledge of the computer from—Turing taught him.

A few weeks after Baby ran its first program, Turing accepted the offer of a job at Manchester University. At last he could get his hands on a universal Turing machine in hardware. Turing improved on the bare-bones facilities, designing an input-output system based on wartime equipment used at Bletchley Park. Williams and Kilburn themselves knew nothing of Bletchley Park and its nine gigantic Colossus computers.[12] The ultra-secret Colossus was the world's first large-scale electronic computer, although it was not all-purpose and did not incorporate Turing's stored-program concept.[13] Turing based his input-output system for the Manchester computer on the same teleprinter tape that ran through Colossus.[14] His tape reader converted the patterns of holes punched across the tape into electrical pulses, and fed these pulses to the computer. The reader incorporated a row of light-sensitive cells which read the holes in the moving tape—exactly the same technology Colossus had used.

As the months passed, a large-scale computer took shape in the Manchester Computing Machine Laboratory (Fig. 8.2). Turing called it the Manchester Electronic Computer Mark I.[15] A broad division of labor developed that saw Kilburn and Williams working on the hardware, and Turing on the software. Williams concentrated his efforts on developing a new form of supplementary memory, a rotating magnetic drum, while Kilburn took the leading role in developing the computer proper. Turing designed the Mark I's programming system, and went on to write the world's first programming manual.[16] The Mark I was operational in April 1949,

[8] Williams in interview with Evans, 1976 (transcription by Copeland (1997)).

[9] For further information see Copeland (2011a, b), and (2012) Chap. 9.

[10] Letter from Brian Napper to Copeland, 16 June 2002.

[11] Bowker and Giordano (1993), p. 19. Copeland is grateful to Napper for drawing this passage to his attention, in correspondence during 2002.

[12] Kilburn: "I didn't know anything about the work at Bletchley". Kilburn in interview with Christopher Evans in 1976; also Williams in interview with Christopher Evans in 1976. ("The Pioneers of Computing: An Oral History of Computing", London: Science Museum. © Board of Trustees of the Science Museum. Transcription by Copeland.)

[13] Copeland et al. (2006), esp. Chap. 9.

[14] For additional detail see Copeland (2011b), pp. 31–32.

[15] Turing, A. M. (c. 1950), p. 85.

[16] Turing (c. 1950).

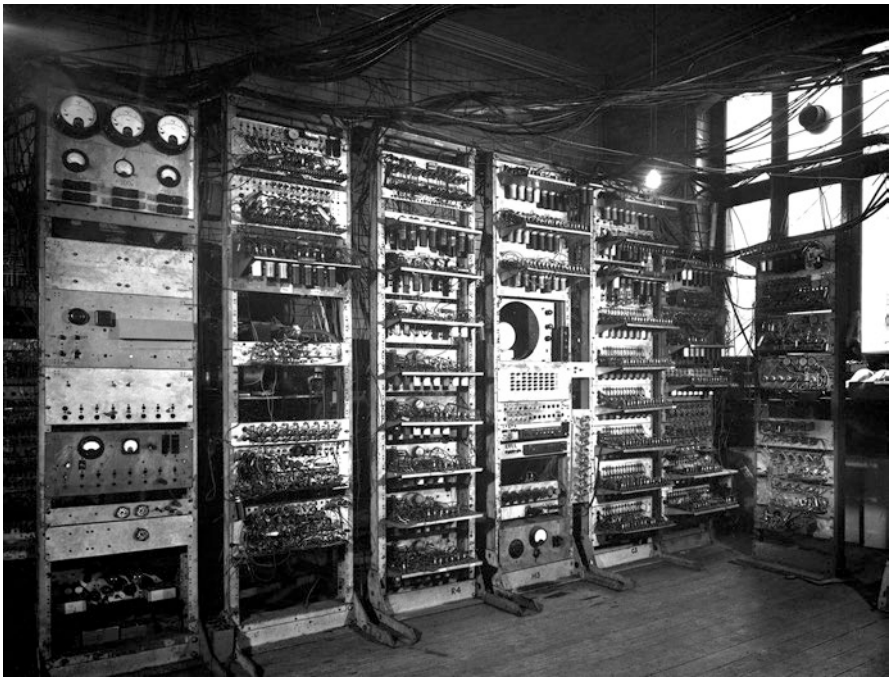


Fig. 8.2 Baby grows into the Mark I (Permission: University of Manchester School of Computer Science)

although additional development continued as the year progressed.[17] Ferranti, a Manchester engineering firm, contracted to build a marketable version of the computer, and the basic designs for the new machine were handed over to Ferranti in July 1949.[18] The first Ferranti computer was installed in Turing's Computing Machine Laboratory in February 1951, a few weeks before the earliest American-built marketable computer became available, the UNIVAC I (Fig. 8.3).[19]

Turing referred to the new machine as the Manchester Electronic Computer Mark II, while others called it the Ferranti Mark I. Turing's nomenclature will be followed here. His programming manual was written in anticipation of the Mark II's arrival, and is titled *Programmers' Handbook for Manchester Electronic Computer Mark II* (c. 1950), but it was the outcome of his programming design work undertaken on the Mark I.[20]

[17] Copeland and Sommaruga (2015), pp. 99–100; Williams and Kilburn (1952).

[18] Williams and Kilburn (1952), p. 59.

[19] The delivery date of the first Ferranti computer is given in a letter from Turing to Woodger, undated, received 12 February 1951 (in the Woodger Papers). A digital facsimile is in *The Turing Archive for the History of Computing* at www.AlanTuring.net/turing_woodger_feb51. For details of the UNIVAC see Stern (1979), p. 17; and Stern (1981), p. 149.

[20] See Turing's preface to (c. 1950).

Fig. 8.3 Turing at the console of the Mark II computer (Permission: University of Manchester School of Computer Science)

Turing's *Handbook* contains what is, so far as is known, the earliest written tutorial on how to program an electronic computer to play musical notes.

8.3 Programming Notes

The Manchester computer had a loudspeaker—the "hooter", it was called—that served as an alarm to call the operator when the machine needed attention.[21] With some simple programming, the loudspeaker could be made to emit musical notes.

The computer's "hoot instruction" worked like this. There was an electronic clock in the computer synchronizing all the operations. This clock beat steadily, like a silent metronome, at a rate of thousands of noiseless ticks per second. Executing the hoot instruction a single time caused a sound to be emitted at the loudspeaker, but the sound lasted no longer than a tick, a tiny fraction of a second. Turing described this sound as "something between a tap, a click, and a thump".[22] Executing the hoot instruction over and over again resulted in this brief sound being produced repeatedly, on every fourth tick: tick tick tick *click*, tick tick tick *click*.[23]

[21] There is a circuit diagram of the hooter in Dodd (c. 1953), Diagram 10.

[22] Turing (c. 1950), p. 24.

[23] Dodd (c. 1953), p. 59.

If the clicks are repeated often enough, the human ear no longer hears discrete clicks but a steady note. Turing realized that if the hoot instruction is repeated not simply over and over again, but in different patterns, then the ear hears different musical notes. For example, if the pattern tick tick tick *click*, tick tick tick tick, tick tick tick *click*, tick tick tick tick is repeated, the note of C_5 is heard. (The subscripted number indicates the octave in which the note occurs. Turing described C_5 as middle C, as musicians sometimes do, especially if playing an instrument with a very high register; however, it is more usual to call C_4, which is an octave below C_5, middle C.[24]) Repeating the different pattern tick tick tick *click*, tick tick tick *click*, tick tick tick tick, tick tick tick *click*, tick tick tick *click*, tick tick tick tick produces the note of F_4—and so on. It was a wonderful discovery.

Turing himself seems not to have been particularly interested in programming the machine to play conventional pieces of music. The different musical notes were used as indicators of the computer's internal state—one note for "job finished", others for "error when transferring data from the magnetic drum", "digits overflowing in memory", and so on.[25] Running one of Turing's programs must have been a noisy business, with different musical notes and rhythms of clicks enabling the user to "listen in" (as Turing put it) to what the program was doing. He left it to someone else, though, to program the first complete piece of music.

8.4 God Save the King

One day Christopher Strachey turned up at the Computing Machine Laboratory (Fig. 8.4). Before the war, he had known Turing at King's College, Cambridge. Strachey was soon to emerge as one of Britain's most talented programmers, and he would eventually direct Oxford University's Programming Research Group. When he first strode into the Manchester Computing Machine Laboratory he was a mathematics and physics master at Harrow, one of Britain's foremost schools. Strachey felt drawn to digital computers as soon as he heard about them, in about January 1951, and taking the bull by the horns he wrote to Turing in April.[26] Turing sent a copy of his *Handbook* (c. 1950) and Strachey studied it assiduously.[27] This was "famed in those days for its incomprehensibility", Strachey said.[28] An ardent pia-

[24] By the time of the third edition of Turing's *Programmers' Handbook* (prepared by Tony Brooker in 1953), Turing's "about middle C" had been replaced by "an octave above middle C".

[25] Prinz (1952), section 20. Copeland is grateful to Dani Prinz for supplying a cover sheet that shows the date of this document.

[26] Letter from Strachey to Max Newman, 5 October 1951 (in the Christopher Strachey Papers, Bodleian Library, Oxford, folder A39); letter from Strachey to Michael Woodger, 13 May 1951 (in the Woodger Papers).

[27] Letter from Strachey to Newman, 5 October 1951. Strachey's copy of Turing's *Programmers' Handbook* still exists, signed on the cover "With the compliments of A. M. Turing" (in the Christopher Strachey Papers, folder C40).

[28] Foy (1974), p. 10.

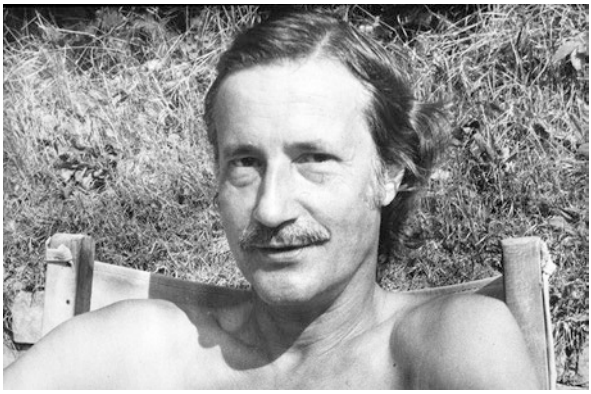


Fig. 8.4 Christopher Strachey sunbathing in the garden of his cottage "The Mud House"; the photo was taken in 1973, two years before his untimely death (Courtesy of the Bodleian Library and Camphill Village Trust)

nist, he appreciated the potential of Turing's terse directions on how to program musical notes. Strachey first visited the Computing Machine Laboratory in July 1951; Turing decided to drop him in at the deep end and suggested he try writing a program to make the computer check itself.[29] When Strachey left the Laboratory, Turing turned to his friend Robin Gandy and said impishly, "That will keep *him* busy!"[30]

It did keep him busy, during the school summer holidays of 1951.[31] Strachey was a precocious programmer and when he "trotted back to Manchester", he recollected, he had with him twenty or so pages covered in lines of programming code—at that time by far the longest program to be attempted.[32] "Turing came in and gave me a typical high-speed, high-pitched description of how to use the machine", Strachey recounted.[33] Then he was left alone at the computer's console until the following morning.

"I sat in front of this enormous machine", Strachey said, "with four or five rows of twenty switches and things, in a room that felt like the control room of a battleship."[34] It was the first of a lifetime of all-night programming sessions. He worked on debugging his monster program, which he called "Checksheet".[35] The name was a variation on a term Turing had used, in his *Programmers' Handbook*, for a hand method of checking programs. Turing called his method "Check Sheets". The

[29] Letter from Strachey to Newman, 5 October 1951; Robin Gandy in interview with Copeland, October 1995.

[30] Gandy in interview with Copeland, October 1995.

[31] Strachey in Foy (1974), p. 11.

[32] Strachey in Foy (1974), p. 11.

[33] Strachey in Foy (1974), p. 11.

[34] Strachey in Foy (1974), p. 11.

[35] Strachey gave the name of the program in his letter to Newman, 5 October 1951. The Checksheet program itself is in the Christopher Strachey Papers (folder C52).

method was "done on paper with quarter inch squares on which vertical lines are ruled in ink", Turing explained in the *Handbook*.[36]

As well as spending the night struggling to debug Checksheet, Strachey prepared a surprise. He managed to debug and get running another program that he'd brought with him. To the astonishment of onlookers, the computer raucously hooted out the British National Anthem.[37] A budding programmer could hardly have thought of a better way to get attention. A few weeks later, Max Newman, Professor of Mathematics at Manchester and founder of the Computing Machine Laboratory, heard the computer grinding out "God Save the King". Newman quickly wrote a letter to Strachey suggesting he might like a programming job in the Lab.[38]

Manchester's musical computer also caught the attention of the popular press, with headlines like "Electronic brain can sing now".[39] The accompanying article explained that "the world's most powerful brain" was "given a coded version of the score", from which it "constructed the necessary waveform". The BBC sent a recording team together with a radio presenter from *Children's Hour*, known as Auntie, to capture a performance by the computer.[40] As well as "God Save the King", the BBC recorded a version of Glenn Miller's "In the Mood", a reedy and wooden performance of the famous hit. There was also an endearing, if rather brash, rendition of the nursery rhyme "Baa Baa Black Sheep". The Mark II, still full of glitches, managed to crash in the middle of its Glenn Miller party piece. "The machine's obviously not in the mood", Auntie gushed.

The unedited BBC recording of the session conveys a sense of people interacting with something entirely new. "The machine *resented* that", Auntie observed at one point. The idea of a thinking machine, an electronic brain, was in the air at Manchester. Turing merrily fanned the flames. He provocatively told a reporter from *The Times* that he saw no reason why the computer should not "enter any one of the fields normally covered by the human intellect, and eventually compete on equal terms".[41]

Max Newman lectured on the new computer music in 1952, to 250 professional musicians who were attending the annual conference of the Incorporated Society of

[36] Turing (c. 1950), p. 12.

[37] Frank Cooper in interview with Chris Burton in 1994; an audio recording of part of the interview is at http://curation.cs.manchester.ac.uk/digital60/www.digital60.org/media/interview_frank_cooper/index-2.html. In the secondary literature it is sometimes said that "God Save the King" was played at the end of Strachey's draughts (checkers) program, but this is not correct (see e.g. Link (2012/2013), p. 23). For further information about Strachey's draughts program, see Copeland (2012), Chap. 9.

[38] Letter from Newman to Strachey, 2 October 1951 (in the Christopher Strachey Papers, folder A39).

[39] See "Electronic Brain Can Sing Now". *The Courier and Advertiser*, 28 February 1952. We are grateful to Diane Proudfoot for finding this article and supplying us with it.

[40] Cooper interviewed by Burton.

[41] Turing quoted in "The Mechanical Brain", *The Times*, 11 June 1949.

Musicians. His lecture was reported in the national press.[42] After explaining that, to make the Manchester computer play melodies, "All you have to do is to send an instruction to the hooter with the frequency of the note you want it to play", Newman described the discovery that the computer could be programmed to *compose* tunes for itself. So far these were, he admitted, "very bad tunes". (Quite possibly the program used Turing's random number generator, a standard hardware component of the Ferranti computers.) According to the *Manchester Guardian*:

> The next step, said Professor Newman, would be to make a machine which could compose *good* tunes, but so far no method of bridging the gap had been devised.[43]

The article continued:

> Professor Newman ended with this note of comfort for the assembled musicians: "All this appears much more alarming and dangerous than it really is. When you see how it is done and how far it is from genuine composition, composers will realise they need not start taking steps to protect themselves against competition from machines."

8.5 Turing's Music Tutorial

Turing's brief tutorial in his *Handbook* was typically compressed and demanding; yet, equally typically, his terse account told readers everything it was necessary to know in order to start writing note-playing programs. Turing called the hoot instruction /V, pronounced "slash vee". The complete tutorial occupied little more than half a page:

> The hooter. When an instruction with function symbol /V is obeyed an impulse is applied to the diaphragm of a loudspeaker. By doing this repeatedly and rhythmically a steady note, rich in harmonics, can be produced. This is used to enable the operator to be called to attend to the machine in some way. The simplest case is where the whole of a job is completed and it is required to clear the electronic stores and start something different. All that is then required is to repeat a cycle of instructions including a hoot, e.g.
>
> ```
> FS  NS/V
> CS  FS/P
> ```
>
> In this case every second instruction will put a pulse into the speaker. These pulses will occur at intervals of 8 beats i.e. 1.92 ms giving a frequency of 521 cycles (about middle C). Or one could use the loop of three instructions

[42] "Very Bad Tunes", *Manchester Guardian*, 4 January 1952. We are grateful to Diane Proudfoot for finding this article and supplying us with it.

[43] "Very Bad Tunes", emphasis added.

```
O@        /V
G@      P@/V
M@      O@/P [see our footnote][44]
```

> which gives a slightly louder hoot a fifth lower in frequency. Single pulses applied to the loudspeaker are distinctly audible as something between a tap, a click, and a thump. This fact can be turned to good account. By putting hoot instructions into programmes at suitable points one is enabled to "listen in" to the progress of the routine. Some indication of what is going on is given by the rhythm of the clicks that are heard.[45]

In these two loops, one consisting of two instructions and one consisting of three, Turing has used international teleprinter code to abbreviate the instructions. At the level of "machine code", the instructions consist simply of strings of binary digits (bits). Teleprinter code associates keyboard characters with strings of 5 bits; for example, *A* is 11000 and *B* is 10011. Teleprinter code was well known to engineers in that era, and was very familiar to Turing from his wartime work at Bletchley Park on the "Tunny" teleprinter code, used by Hitler and his generals. To Turing, teleprinter code must have seemed a natural choice for abbreviating the Manchester computer's bitcode. This system's main defect, that the abbreviations give no intuitive sense at all of what is being abbreviated, is one reason why his *Handbook* was such heavy going.

Let us unpack what Turing wrote down, in order to clarify its importance in the history of computer music. First, we explain Turing's notationally formidable subroutines, simplifying his notation and spelling out the connection between his sequences of coded instructions and perceived musical sounds. Then we report the results of our computer-assisted analyses of the BBC's 1951 recording of the Manchester computer's music. From these results, we extract our account of how others at Manchester extended Turing's note-playing subroutines, so enabling the Mark II to play its first melodies. We also provide a series of tables that allow the

[44] For ease of exposition, we have replaced Turing's G@/P by O@/P, thereby oversimplifying the behaviour of the /P instruction, but making the loop ostensibly easier to follow. Appearances to the contrary notwithstanding, Turing's G@/P does take execution back to the start of the loop, while our oversimplified version does not do so. For a full explanation of /P, see Prinz (1952), p. 14. Strachey, who marked corrections by hand on his copy of Turing's *Handbook*, altered this loop to:

```
O@   /V
B@   Q@/V
G@   B@/P
```

See also Strachey's typed sheets of errata to Turing's *Handbook* dated 9 July 1951; in the Christopher Strachey Papers, folder C45.

[45] Turing (c. 1950), p. 24. There is a magisterial introduction to programming (what Turing called) the Mark II in Campbell-Kelly (1980).

reader to see not only how these melodies were produced but also, more generally, how the computer itself can be used in historical detective work *about* computers.

In teleprinter code, / is 00000 and V is 01111; thus /V is the teleprinter code abbreviation of the Mark II's 10-digit hoot instruction, 0000001111. Turing's /P is also an instruction; instructions always began with /, or T (00001). The other symbols in Turing's two sample subroutines, NS, P@, FS, CS, O@, G@ and M@, are memory addresses: each pair of symbols abbreviates a 10-digit address in the computer's Williams tube memory.

Instruction /P (unconditional transfer of control) tells the machine to obey next the instruction stored at a location specified via the address immediately to the left of the /. In effect the second line of the first loop sends the machine back to the first line (but see our footnote); and the final line of the second loop again sends the machine back to the first line.[46] The computer will continue to loop until an instruction from elsewhere in the program terminates the loop after *n* repetitions.[47] The programmer selects the number *n*, so determining how long the note is held, as required by the rhythms of the melodies.

Our analysis of the BBC's recording of the Mark II playing "God Save the King", "Baa Baa Black Sheep", and "In the Mood", showed that the durations of the played notes varied between 80 milliseconds and 1100 milliseconds. The analysis also revealed that very short pauses were programmed between each consecutive note, presumably by means of "silent" loops—short loops containing no hoot instruction. The duration of these inter-note pauses is between 40 and 50 milliseconds (exact measurement was difficult because the pauses are blurred over by reverberation from the room housing the computer). The inter-note pauses help to define the beginning of each note, and are essential if a sequence of several notes of the same pitch is played. Without a gap between the individual notes, a single long note would be heard.

The occurrence of NS to the left of /V in Turing's first subroutine can be ignored for the present purposes, and so can the P@ to the left of /V in the second subroutine. These terms create special effects to do with the computer's visual display, and have no role in the production of musical notes. The effect produced by including the address NS in line 1 of the first subroutine is that the information stored at NS momentarily brightens on the monitor display as the machine hoots.[48] This provides a visual prompt to assist the operator. Similarly, the effect of P@ in the second instruction of the three-line subroutine is to cause the information stored at P@ to brighten on the display as the hooter sounds. The two note-playing subroutines can be written more cleanly without these special effects:

[46] See note 44.

[47] Turing explains loop control (by means of a B-tube) in his *Handbook*, pp. 66–67; the B-tube was effectively a register containing *n* and this number was counted down by repeatedly subtracting 1.

[48] Turing (c. 1950), p. 22.

```
FS     /V
CS    FS/P

O@     /V
G@     /V
M@   O@/P
```

Taking clarity a step further, we might replace the teleprinter-coded addresses with simple line-numbers:

```
1    /V
2   1/P

1    /V
2    /V
3   1/P
```

As Sect. 8.3 mentioned, the /V instruction takes four ticks to complete—four *beats* in the Manchester jargon—with the actual hoot occurring on the fourth beat.[49] /P also takes four beats to complete. As Williams and Kilburn put it, the basic rhythm of the Manchester computer was "four beats to the bar".[50] Thus, running through Turing's two-line subroutine once produces: tick tick tick *click*, tick tick tick tick; and looping repeatedly through the subroutine produces the first of the two sequences discussed in Sect. 8.3. Similarly, running through the three-line subroutine once gives: tick tick tick *click*, tick tick tick *click*, tick tick tick tick; and looping repeatedly gives the second sequence in Sect. 8.3.

The precise duration of a single beat was 0.24 milliseconds (ms). The first subroutine produces one click every 8 beats, which is to say every 1.92 ms. Thus, the frequency with which clicks are produced, as the machine loops repeatedly through the subroutine, is (1 ÷ 1.92) clicks per ms—0.52183 clicks per ms, or 521.83 clicks per second. In standard units, the frequency of the clicks is said to be 521.83 Hertz (Hz). This is close to C_5, whose assigned frequency in the 'Equal Tempered' scale is 523.25 Hz. The Equal Tempered scale is the standard scale for keyboard instruments, with adjacent keys playing notes heard as equidistant from one another.[51]

Table 8.1 shows the Equal Tempered frequencies of all the notes occurring in the fragments of the scores of "God Save the King", "In the Mood" and "Baa Baa Black Sheep" that were performed in the BBC recording. Later tables show the actual frequencies that the computer produced.

[49] The Mark II was synchronised by an oscillator with a frequency of 100 kHz. Turing called a single cycle of the oscillator the "digit period". The digit period was 10-microseconds and the duration of a beat was 24 digit periods.

[50] Williams and Kilburn (1952), p. 57.

[51] We follow the A = 440 Hz tuning standard.

Table 8.1 The Equal Tempered frequencies of the notes from those parts of the scores of "God Save the King", "In the Mood" and "Baa Baa Black Sheep" that were performed in the 1951 BBC recording

Note	Frequency (in Hertz)
$F\#_2$	92.5
G_2	98
A_2	110
B_2	123.47
C_3	130.81
$C\#_3$	138.59
D_3	146.83
E_3	164.81
$F\#_3$	185
G_3	196
A_3	220

Later tables show the measured frequencies that the computer produced

By dispensing with any reference to memory addresses, and abstracting from which particular instructions are employed, Turing's note-playing subroutines can be represented very transparently by means of what we call note-loops. The note-loop corresponding to Turing's C_5 routine is:

START – – – H – – – –REPEAT

Each "–" represents a single beat, with "H", the hoot, occurring on the fourth beat of the first bar. Representing this more economically still, the note-loop is simply:

$$<3\text{H}, 4>.$$

8.6 Exploring the Mark II Notes

Subroutines for playing lower notes require the addition of further instructions, since this has the effect of adding extra blocks of 4 beats between the hoots, so lowering the frequency. Conveniently, one of Turing's instructions, /L, served to waste time: its execution took up 4 beats but it did nothing. (Strictly, the instruction did nothing unless a "dummy stop" switch had been set manually at the control console before the program started, in which case /L caused the machine to pause. We discuss dummy stops in Sect. 8.8.) /L is ideal for creating lower-frequency notes.[52] For example, the note-loop <3H, 4, 4> produces a frequency of 347.22 Hz, approximately F_4 (349.23 Hz), a fifth lower than C_5.

[52] /L's companion dummy-stop instruction /G serves just as well, as do so-called "dummy" instructions such as T£, TM, and TX: these have no effect except to cause a delay of 4 beats (Prinz (1952), p. 20; Turing (c. 1950), p. 58 and Fig. E.).

<3H, 4, 4> produces the same note as Turing's second example of a loop, which in our notation is <3H, 3H, 4>. Adding the second pulse of sound at the same frequency does not alter the note, but (as Turing said) has the effect of making the note louder. We call note-loops that play the same frequency *equivalent*.

Some further examples of note-loops are <3H, 4, 4, 4, 4, 4, 4, 4>, producing a frequency of 130.21 Hz, fairly close to C_3 (130.81 Hz), and <3H, 4, 4, 4, 4, 4, 4>, producing a frequency of 148.81 Hz, lying between D_3 (146.83 Hz) and D_3 sharp (155.56 Hz). Both these note-loops produce a quiet sound. The same notes are played more loudly if extra hoots are added to form equivalent note-loops, such as <3H, 3H, 3H, 3H, 4, 4, 4, 4> and <3H, 3H, 3H, 3H, 4, 4, 4> respectively. We call a note-loop containing only one hoot the *primary* form, and equivalent note-loops containing more than one hoot *padded* forms of the loop. Padded note-loops typically produce notes with a different timbre or tone-color from the note produced by the loop's primary form. Timbre is manifested by differences in the shape of waveforms of the same frequency. (If a violin and a flute play exactly the same note at exactly the same volume, the sounds are nevertheless instantly recognizable as different, because of their different timbres.)

We built a simulator in order to investigate the effects of padding note-loops, and also to establish that our calculated note-loops really do play the correct notes. Our simulations of the Mark II playing "God Save the King", "In the Mood" and "Baa Baa Black Sheep" can be heard at www.AlanTuring.net/MarkII_music_simulations.mp3. An Atmel ATmega168 microcontroller was used to create a functional computer simulation of the Mark II as a note-playing device. We connected a small loudspeaker directly to one of the digital output pins. Microcontroller programs using pulses and delays reproduced the beat-structure of the Mark II and emulated the effects of the Mark II's music routines. We found that primary note-loops produce relatively thin-sounding notes while their padded equivalents produce somewhat louder, fuller-sounding notes. Over-padding is possible, however. The simulator revealed that including too many hoots adds a high overtone, especially with lower notes containing more beats. Because an uninterrupted sequence of hoot instructions generates the Mark II's highest achievable note of 1041.67 Hz (somewhere in the vicinity of C_6), the result of over-padding a note-loop is that the ear tends to hear not only the intended note but also this maximum note as a high overtone.

The BBC recording indicates that the programmer most likely used padding. If only unpadded loops are used, lower notes are quieter than higher notes, since in a lower note there are longer gaps between the hoots. This is not observed in the recording, and in fact some lower notes are louder than some higher notes. However, because of the poor quality of the recorded material, the analysis described here did not reveal the number of hoots used in each individual note-loop. Our reconstruction of the note-loops in Sect. 8.12 retrieves the primary form of the note-loops only.

Although the normal rhythm of the Manchester computer was 4 beats to the bar, some instructions took 5 beats to execute. Incorporating a suitable 5-beat instruction in note-loops (e.g. Turing's instruction TN[53]) extends the number of playable notes.

[53] Turing (c. 1950), Fig. E.

Table 8.2 The frequencies that the Manchester Mark II was able to play by means of loops containing 4-beat instructions or mixtures of 4- and 5-beat instructions (down to the lowest frequency in the 1951 BBC recording)

Beats	Frequency (in Hertz)
8	520.83
12	347.22
13	320.51
16	260.42
17	245.10
18	231.48
20	208.33
21	198.41
22	189.39
23	181.16
24	173.61
25	166.67
26	160.26
27	154.32
28	148.81
29	143.68
30	138.89
31	134.41
32	130.21
33	126.26
34	122.55
35	119.05
36	115.74
37	112.61
38	109.65
39	106.84
40	104.17
41	101.63
42	99.20
43	96.90

By increasing the number of beats in a loop still further, the machine will play ever lower notes, until at approximately 20 Hz the human ear begins to perceive a series of individual clicks rather than a note

For example, adding 10 extra beats to either the primary or the padded form of the 148.81 Hz note-loop displayed previously results in a loop that plays 109.65 Hz, very close to A_2 (110 Hz); the primary form is <3H, 4, 4, 4, 4, 4, 4, 5, 5>. The following loop plays the low note F_2 sharp ($F\#_2$): <3H, 4, 4, 4, 4, 5, 5, 5, 5, 5>. This loop produces 92.59 Hz, fractionally higher than the note's Equal Tempered frequency of 92.5 Hz.

In what follows, note-loops are sometimes written in an abbreviated form. For example, <3H, 4×7> replaces <3H, 4, 4, 4, 4, 4, 4, 4> and <3H, 4×4, 5×5> replaces <3H, 4, 4, 4, 4, 5, 5, 5, 5, 5>. Table 8.2 shows the full range of frequencies, down to 96.9 Hz, that the Mark II could produce by means of note-loops containing 4- or 5-beat instructions.

8.7 Hoot-Stop and Radio Stop

As Turing explained in his tutorial, a fundamental use for <3H, 4> was the so-called "hoot-stop". If the two lines of code displayed in the tutorial were placed at the end of a routine (or program, as we would say today), then once the routine finished running, the computer would sound C_5 continuously until the operator intervened. The intervention might take the form of pressing the "KEC" key at the control console—the "clear everything" key—in order to clear out the completed routine, in preparation for running the next.[54] Without the convenient hoot-stop facility, the operator was obliged to remain at the console, watching the indicators, in order to tell whether the routine had stopped running.

A very different solution to effectively the same problem was found in the case of BINAC, an early US computer. Herman Lukoff, one of BINAC's engineers, explained that the technician whose job it was to monitor BINAC through the night, Jack Silver, had to spend all his time "looking at the flashing lights; it was the only way of knowing that the computer was working". One night Silver switched on a radio to alleviate the monotony:

> To Jack's surprise, all kinds of weird noises emanated from the loudspeaker, instead of soothing music. He soon realized that the churning BINAC generated these noises because as soon as it halted, the noises stopped. … He put the computer-generated tones to good use. Jack found that by turning the volume up he was able to walk around the building and yet be immediately aware of any computer stoppage.[55]

BINAC, built in Philadelphia by Presper Eckert, John Mauchly, and their engineers at the Eckert-Mauchly Computer Corporation, was the stored-program successor to the pioneering Eckert-Mauchly ENIAC. Eckert and Mauchly went on to build UNIVAC, one of the earliest electronic digital computers to enter the marketplace.

The first systematic use of the Manchester Mark I's programmable hooter appears to have been to provide the hoot-stop facility.

[54] Dodd (c. 1953), p. 32.

[55] Lukoff (1979), pp. 85–86.

8.8 First Hoots

In a section of his *Handbook* devoted exclusively to the Mark I machine, Turing made it clear that the programmable hooter predated the Mark II machine.[56] /V was Mark II notation; in the Mark I era it was the instruction K (11110) that caused the loudspeaker to sound. The Mark I, which was closed down in the summer of 1950,[57] was a slower machine than the Mark II; the duration of a beat was 0.45 ms, compared with the Mark II's 0.24 ms. This considerably reduced the number of playable notes. For example, lengthening the beat to 0.45 ms causes the frequency of <3H, 4> (the highest-frequency loop) to drop from 523.25 Hz to 277.78 Hz, approximately $C\#_4$.

It is not known precisely when a programmable hooter was first added to the computer. Geoff Tootill's laboratory notebook is one of the few surviving documents relating to the transition from Baby to Mark I.[58] In a notebook entry dated 27 October 1948, Tootill listed the K instruction 11110 among the machine's 32 instructions, but indicated that it was unassigned at this time. Given Turing's focus on the programming side, and the emphasis he placed on the use of the hoot instruction and pause-instructions, which he called "dummy stops", for "testing"—i.e. debugging—new routines, it seems likely that the hooter was incorporated earlier rather than later.[59] The computer was running complex routines by April 1949, in particular a routine that searched for Mersenne primes (primes of the form 2^n-1).[60] Most likely Turing's debugging toolkit of hoots and dummy stops was introduced earlier than this. It also seems likely that the use of the K instruction, and the use of loops to increase the volume of the hooter's native clicks, probably occurred more or less simultaneously. The loud note produced by the loop would have been more useful than the quiet click given by a single instruction. As Dietrich Prinz, a regular user of the Mark I, said, "By programming a simple loop containing this instruction … an *audible* 'hoot' is emitted".[61]

A table in Tootill's notebook dated 28 November 1948, showing the machine's instructions at that time, listed three different dummy stops, N, F, and C. The section of Turing's *Handbook* dealing with the Mark I explained that, during the checking of a program, the dummy stops N, F, and C would be operated in conjunction with the hoot instruction K. By the time of the 28 November table, the K instruction had been assigned: Tootill listed its function as "Stop". However, his table also contains another instruction labeled "Stop" (00010). Since the machine had no need of two ordinary stop-instructions, it seems very likely that K was being used for hoot-stop at this time. When execution of the program reached the point where the hoot-stop

[56] Turing (c. 1950), pp. 87–88.

[57] Williams and Kilburn (1952), p. 59.

[58] Tootill (1948–9), table of the machine's instructions dated 27/10/48.

[59] Turing (c. 1950), pp. 24, 88.

[60] Lavington (1980), p. 37.

[61] Prinz (1952), p. 20 (our italics).

had been inserted, execution would pause and the hooter would play the note of $C\#_4$ (middle C sharp) continuously until the operator intervened. We conclude that the Mark I was playing at least one note in about November 1948.

8.9 Other Early Music

The Manchester Mark I was not the only zeroth-generation electronic stored-program computer to play music. Trevor Pearcey's Sydney-built CSIRAC (pronounced "sigh-rack") had a repertoire that included "Colonel Bogey", "Auld Lang Syne", and "The Girl with the Flaxen Hair", as well as brief extracts from Handel and Chopin. Some of the music routines survived on punched paper tape, but seemingly no audio recordings were preserved. Australian composer Paul Doornbusch has recreated some of the music, using reconstructed CSIRAC hardware and the surviving programs.[62] CSIRAC, still complete and almost in working order, is in Melbourne Museum.

Doornbusch's recordings and the BBC's Manchester recording show that the programmers of both computers ran into the problem of "unplayable notes"—notes that could not be replicated or even closely approximated by means of an available note-loop. An example is the note of D_3, which occurs 5 times in the BBC recording of "God Save the King". This note's Equal Tempered frequency is 146.8 Hz, but the closest that the Mark II can approach is the significantly different note of 148.81 Hz, discussed in Sect. 8.6. To judge from the Doornbusch recordings, $F\#_2$, G_2, $C\#_3$, $F\#_3$, D_4, E_4, F_4, G_4, and A_4 were particularly troublesome for CSIRAC. The Australian and British solutions to the problem of unplayable notes were distinctively different. The Manchester programmers opted to use the nearest playable frequency and tolerated the melody being less in tune (see Sect. 8.14 for a fuller discussion of this technique). CSIRAC's programmers, on the other hand, attempted to mimic the unplayable frequency by rapidly moving back and forth between two playable frequencies that bracketed the note in question. The result was a melody in which tuning-related problems were replaced by timbre-related problems, with the Australian technique producing notes that sound grainy and unnatural.

An embryonic CSIRAC first ran a test program in about November 1949.[63] The computer seems to have been partially operational from late 1950, and in regular operation from about mid 1951. The date when CSIRAC first played musical notes is unrecorded; presumably this was in late 1950 or in 1951. The computer is known to have belted out tunes at the first Australian Conference on Automatic Computing Machines, held at Sydney University in August 1951.[64] A 2008 BBC News article, based on Australian sources, stated that CSIRAC was the first computer to play

[62] Doornbusch (2005). The book includes a CD of recreated music.

[63] McCann and Thorne (2000), p. 2.

[64] McCann and Thorne (2000), p. 3; Doornbusch (2005), pp. 24–25.

music.[65] The only evidence offered for this claim was that CSIRAC's performance at the Sydney Conference allegedly preceded the date of the BBC recording of the Manchester computer (the recording described in Sect. 8.4). However, the date of the BBC recording is in fact unknown; and in any case the Manchester computer's first performance of "God Save the King"—whose precise date is also unknown—would have occurred some days or weeks or even months before the BBC recording was made. Unfortunately, an Australian contribution to *The Oxford Handbook of Computer Music* also states, without evidence, that CSIRAC was "the first computer to play music".[66] CSIRAC, however, was certainly not the first computer to play music.

There were American hoots too. BINAC was playing music before CSIRAC even ran its first test program. BINAC was completed in August 1949 (although it ran a 50-line test program in April of that year).[67] As Lukoff explained, a party was held to celebrate the machine's completion:

> It was held right at the BINAC test area one August evening. In addition to hors d'oeuvres and cocktails, the BINAC crew arranged a spectacular computer show. Someone had discovered that, by programming the right number of cycles, a predictable tone could be produced. So BINAC was outfitted with a loudspeaker attached to the high speed data bus and tunes were played for the first time by program control. The audience was delighted and it never occurred to anyone that the use of a complex digital computer to generate simple tones was ridiculous. … The crowning achievement of the evening came after a long, laborious arithmetic computation; the machine laid an egg! The engineers had programmed the machine to release a hard-boiled egg from its innards.[68]

As far as can be ascertained, therefore, the first melodies to be played by a computer were heard at the Eckert-Mauchly Computer Corporation in the summer of 1949, and very likely individual musical notes were heard a few months earlier at Turing's Computing Machine Laboratory, probably in November 1948. Effectively, the pioneering developments on either side of the Atlantic were roughly contemporaneous, with Australia entering the field a year or two later.

8.10 The BBC Recording

The BBC's website offers an edited digitized version of the original BBC recording of the Manchester Mark II; and there is a full-length version of the same digitalization of the recording on the Manchester University website.[69] Upon pressing play, the listener is greeted by a thick wall of noise—a combination of hissing, humming,

[65] Fildes (2008).

[66] Dean (2009), pp. 558, 584.

[67] Lukoff (1979), p. 84.

[68] Lukoff (1979), p. 86.

[69] Fildes (2008); http://curation.cs.manchester.ac.uk/digital60/www.digital60.org/media/mark_one_digital_music/index.html.

and rhythmically repeating crackles from the original acetate disc. Then a tone not unlike a cello cuts through this cacophony to give a mechanical-sounding rendition of the first two phrases of the National Anthem. The melody, though familiar enough, is somewhat out of tune, with some notes more distinctly out than others. Moreover, some notes are loud relative to their neighbors (most likely the result of padding). At the end of the second phrase, the performance is suddenly cut short by a glitch and nervous laughter.

The engineers restart the routine and this time the machine energetically plays its way through the entire first verse. Then, with scarcely a pause, it follows up with an unbroken performance of the first line of "Baa Baa Black Sheep". For its third number the Mark II attempts "In The Mood", but once again falls victim to an unknown error, causing it to sing out a high pitched beep. The recording team give the computer one more chance to make its way through "In The Mood", and it proceeds admirably until the final line, when it yet again breaks down. Altogether, the recording lasts about 3 minutes.

The frequencies in the recording are higher than the ideal in-tune frequencies that would be heard if the same melodies were played on a modern synthesizer. Because the Mark II offered such a limited palette of notes, deviations such as this are to be expected. In scoring a melody for the Mark II, the musician had to try to ensure that the individual notes were as in tune with one another as possible. Whether or not notes were played at the ideal Equal Tempered frequency would necessarily have been a secondary consideration.

The waveform of the recording looks and sounds very unlike the square pulse-wave produced by a modern synthesizer. The early waveform is much rounder, with a lot less high-frequency energy. Its departure from the ideal square-wave is most likely a result of the Mark II's pioneering digital electronics, together with the long and transformative signal chain between the original sound source and the final digital file. Obtaining clean pulse waves from early digital electronics was a challenge in itself; the pulses from the computer would have resembled trapezoids rather than perfect squares, and so would already be lacking some of the high frequency audio content of modern pulse waves. The hooter, the next element in the chain, may itself have transformed the sound somewhat, depending on the characteristics of the loudspeaker. Then the microphone used to record the performance and the amplifier used to boost the signal, as well as the room in which the performance took place, would all have colored the audio. Finally, the cutting mechanism, which gouged a groove in the rotating acetate disc, was a source of pitch instability and plenty of crackles and noise. Each of these stages provided generous opportunities for the introduction of unwanted noise, artifacts, pitch instability, and spectral alterations.

Two different acetate discs were cut during the recording session. One was taken away by the BBC and presumably used in a broadcast. It is unlikely that this disc survives, but a second disc was given to Manchester engineer Frank Cooper as a souvenir. It contained another recording made at Cooper's request once the main

recording session was over.[70] By that time, Cooper recollected, "the computer was getting a bit sick and didn't want to play for very long". Eventually he donated this 12-inch single-sided acetate disc to the Computer Conservation Society; and subsequently the National Sound Archive, now part of the British Library, made a digital preservation copy of the recording.[71] This further stage of processing may have added its own sonic characteristics.

8.11 The Paradox of the Impossible Notes

Despite the long chain existing between the original music-playing routine and the available recording, we decided to attempt to reverse-engineer the note-loops that the programmer had used. A frequency analysis of the digital recording told us the frequencies of the recorded notes, and where different recorded occurrences of the same note had different frequencies, we were able to gather information about error magnitudes. (The principal software used was iZotope RX for removing noise and artifacts, and Celemony Melodyne for carrying out the frequency analyses and pitch correction.)

Soon, however, we hit a fundamental problem. There were frequencies in the recording that could not possibly have been produced by the Mark II's note-loops—*impossible* notes. An example is the recorded note corresponding to E_3 (which occurs only once in the verse from the National Anthem). The measured frequency of the recorded note is 169.06 Hz, a thumping 4.25 Hz distant from the note's Equal Tempered frequency of 164.81 Hz. 169.06 Hz is an impossible note for the Mark II. It is bracketed by a note-loop of 24 beats, producing a note of 173.61 Hz, and a loop of 25 beats, producing a note of 166.67 Hz (see Table 8.2). Whatever frequency it was that the computer had produced, this was certainly not the frequency we found in the recording on the BBC website.

We decided to move closer to the original recording, assuming that the problem lay in the transformative chain between the source and the digital file we were analyzing; and we obtained from the British Library an identical copy of their archived digital file. The recording we had analysed previously from the BBC website was in fact a digitized version of an *analog* recording that had been made by the National Sound Archive—using a cassette tape and Dolby B noise reduction—at the same time as the digital preservation copy was created.[72] (This cassette recording was subsequently digitized by a technician at Manchester University and put on the Internet as part of Manchester's *Digital 60* celebration of Baby's 60th anniversary.) The digital file as we received it from the British Library did indeed exhibit differences from the previously analysed (cassette-based) recording—but not the differences we had hoped for. The new frequencies we measured were consistently higher

[70] Cooper interviewed by Burton.

[71] British Library Sound Archive reference number H3942.

[72] Information from Chris Burton.

Table 8.3 The measured frequencies of the notes in the National Sound Archive digital copy of the BBC recording, prior to correction

Note	Measured frequency (in Hertz)
$F\#_2$	97.07
G_2	101.78
A_2	113.19
B_2	127.42
C_3	135.74
$C\#_3$	145.90
D_3	156.96
E_3	169.54
$F\#_3$	194.42
G_3	203.56
A_3	241.16

than our previous measurements (indicating that the recording involving the cassette tape ran slower than the British Library's preservation copy). However, the impossible notes were still present, as Table 8.3 shows. E_3 is the most troubling case but the high notes $C\#_3$, D_3, $F\#_3$, and G_3 all stand out as being quite far off the playable frequencies shown in Table 8.2.

Although the software used for our frequency analyses is very reliable, the measured frequencies in Table 8.3 nevertheless have a significant margin of error. This is because of a wobble in the speed of the recording that was most likely introduced by the disc-cutting process. This wobble caused some notes in the recording to bend slightly throughout their duration, and the wobble also resulted in the measured frequency of different occurrences of a note being different from one another. The maximum difference between two readings of the same note from different parts of the recording was one fifth of a semitone (20 cents, in the unit of measurement introduced below). The frequencies shown in Table 8.3 are the midpoints of the range of frequencies measured for the note's various occurrences. However, the margin of error in the measured frequencies created by the wobble (±20 cents) is not large enough to explain the presence of the impossible notes. A more probing analysis was needed.

8.12 Searching for the Right Speed

The difference in speed and consequently in pitch between the two analysed recordings is in fact the clue to the impossible notes. If the speed of the archived recording were itself fast or slow, this could account for the presence of impossible frequencies. For example, if the turntable in the BBC recording van were running slightly too fast as the acetate disc was cut, the frequencies would be shifted systematically

upon playback at the standard speed of 78 rpm.[73] (Achieving speed constancy was a problem with the BBC's standard mobile recording equipment.[74]) Only when we know the true speed S_0 at which the recording should be played can we measure the frequencies actually produced by the computer.

Therefore we set out to determine the true speed of the recording. The basic question was whether some small increase or decrease in the overall speed of the recording would result in frequencies that correlated acceptably with the playable frequencies in Table 8.2. Our initial approach to the question was to nudge the speed up or down a bit and see what frequencies emerged. The result was always the same: some frequencies matched well but other frequencies that had matched at a speed investigated previously now no longer matched. How to get the carpet to fit into all the corners of the room at once? We wrote a command-line program to conduct a brute-force search for the optimum global fit. The program (written in C) incremented the speed of the recording in tiny steps and at each increment calculated the new notes, by appropriately tweaking the notes we had measured. We call the notes calculated at each increment the *calculated* notes, the notes that the computer is actually capable of producing the *playable* notes (shown in Table 8.2), and the notes that we measured from the BBC recording the *measured* notes (shown in Table 8.3).

The exhaustive search commenced at 79.5% of the native speed of the recording, an arbitrary point well below the native speed, and progressed through 80,000 increments to 126% of the native speed. At each step, the program computed the 'distance' between each calculated note and its most closely neighboring playable note. The program averaged these distances to produce Φ_S, the closeness-of-fit parameter for speed S; and then the process repeated, with the program incrementing S by a small fixed amount δ and calculating $\Phi_{S+\delta}$.

Our program expressed the distance between a calculated note and the nearest playable note in musical units called "cents". Cents were also the units of Φ_S. One cent is one-hundredth of the distance between any two adjacent keys on the piano keyboard. To the human ear, the distance between, say, the notes C_4 (261.64 Hz) and C_5 (523.28 Hz), which are one octave apart, is musically the same as the distance between C_5 and C_6 (1046.56 Hz), which are again one octave apart. Working in Hertz, however, these distances are far from the same: the distance between C_4 and C_5 is 261.64 Hz, whereas the distance between C_5 and C_6 is 523.28 Hz. Musically, the distances are the same, but the *frequency* of the notes is doubling. If these distances are expressed in cents, on the other hand, they are indeed identical. Since C_4 and C_5 are 12 notes apart on the keyboard, as are C_5 and C_6, the distance is 1200 cents in each case. In fact, not only distances between notes but also the individual notes themselves can be expressed in cents, e.g. by somewhat arbitrarily taking C_0 (16.35 Hz) to be the zero-point, so that C_1 is 1200 cents, C_2 is 2400 cents, and so on.

The program expressed all values in cents rather than Hertz, a procedure that yields a simpler algorithm, since each increment in speed has a uniform effect on the notes if the units are cents, whereas the formula is more complex if the units are

[73] BBC (1950), p. 26.

[74] BBC (1950), pp. 49, 52.

Table 8.4 The frequencies played by the Mark II once the speed of the 1951 BBC recording is corrected

Note	Frequency (in Hertz)
$F\#_2$	99.21
G_2	104.02
A_2	115.69
B_2	130.23
C_3	138.73
$C\#_3$	149.12
D_3	160.43
E_3	173.29
$F\#_3$	198.71
G_3	208.05
A_3	246.48

Hertz. Moreover, if the notes are expressed in cents, the distance between two notes can be calculated simply by subtracting the smaller value from the larger, whereas matters are more complicated if the notes are expressed in Hertz, as our examples show. A calculated note could be, say, 10 Hz higher than the nearest playable note (100Hz, say), while at a higher speed, a calculated note could be 20 Hz higher than its nearest playable note (say 200 Hz), and yet each of the two notes be equally close to its nearest playable note, since differences in frequency are magnified as the speed becomes higher.

The object of our search, the smallest Φ_S, turned out to be just 37.7 cents, and the associated speed, the calculated true speed S_0, was just 2.2% faster than the native speed of the recording. If the speed of the recording is increased by this small percentage—or, equivalently, if our previous measurements of the notes are corrected by adding 37.7 cents to each measured note—then all the notes lie within 9.67 cents of playable notes, well within the margin of error of ±20 cents. In fact, the situation is even better than this. If A_3 is disregarded, all the notes are within 3.52 cents of playable notes. (There are only four samples of A_3 in the entire recording, all in quick succession at the end of "In the Mood".)

The "impossible" notes have disappeared. They were artifacts, caused by the recording becoming slightly slowed down at some point in the transformative chain—most likely in the cutting of the original acetate disc.

Table 8.4 shows the measured notes after adjustment for the calculated true speed. Having discovered the true frequencies, with a margin of error of only a few cents, we were able to conclude that the routines used to play the versions of "God Save the King", "Baa Baa Black Sheep" and "In the Mood" recorded by the BBC must have used note-loops whose primary forms are shown in Tables 8.5 and 8.6. As Sect. 8.6 explained, the programmer(s) probably used padded forms of the primary loops, but our analysis as described here retrieved only the primary forms.

Table 8.5 The primary note-loops used to play the version of "God Save the King" recorded by the BBC, as indicated by our frequency analysis

Note	Beats	Primary Note-Loop
$F\#_2$	42	<3H, 4×7, 5×2>
G_2	40	<3H, 4×9>
A_2	36	<3H, 4×3, 5×4>
B_2	32	<3H, 4×7>
C_3	30	<3H, 4×4, 5×2>
D_3	26	<3H, 4×3, 5×2>
E_3	24	<3H, 4×5>

The programmer may have used padded forms of the primary loops, as Sect. 8.6 explained, but the quality of the recorded material is too poor for the present analysis to retrieve information about the way the padding was done

Table 8.6 The primary note-loops for additional notes in the melodies "Baa Baa Black Sheep" and "In the Mood"

Note	Beats	Primary Note-Loop
$C\#_3$	28	<3H, 4×6>
$F\#_3$	21	<3H, 4×3, 5>
G_3	20	<3H, 4×4>
A_3	17	<3H, 4×2, 5>

8.13 The Question of Authorship

The question of the authorship of the three routines that played the melodies recorded by the BBC is open. In the wake of Strachey's tour de force a number of people in the computing lab started writing music programs: Cooper related that "everybody got interested—engineers started writing music programs, programmers were writing music programs".[75] Nothing about the BBC recording settles the question of authorship: even the routine that played the National Anthem in the recording may have been a retouched version of Strachey's original.

However, it can at least be said that the programmer(s) of the routines for "Baa Baa Black Sheep" and "In the Mood" used the same key signature as the programmer of "God Save the King"; and also used the same primary loops as those selected for "God Save the King": new loops were introduced only for notes that do not occur in the Anthem. This was so even though some alternative primary loops were available, and in fact it is arguable that some of these choices would have produced frequencies that sounded more in tune (see Sect. 8.14).

We expect that a more refined analysis currently underway will reveal information about the use of padding in the recorded melodies: this may help to clarify aspects of the authorship issue.

[75] Cooper in interview with Burton.

8.14 Tuning Puzzles

There are puzzles about some of the frequencies selected by the programmer(s) and so we end with a discussion of tuning.

While there are always differences between the Equal Tempered frequency of a note and the note actually played by the Mark II, these differences do not produce a general out-of-tune-ness in the performance. Certainly if the computer were playing together with some other instrument that was tuned in accordance with the Equal Tempered scale, the computer would sound conspicuously out of tune. However, with the computer playing solo, what matters more than the absolute frequencies of the notes is the extent to which the performed notes are in tune with one another, and in fact the relative tuning of the notes is quite acceptable, actually closer to the key of G# major than G major, the usual key of the National Anthem. In programming the National Anthem, the programmer seems to have picked frequencies with a view to their overall relationships, rather than trying to hit the Equal Tempered frequencies as closely as possible.

Nevertheless, some notes in the recording are sufficiently out of tune to make anyone with a musical ear cringe. Examples are the sequence of occurrences of D_3 at notes 17 to 20 of "God Save the King" and the final occurrence of D_3 at note 36. These notes would sound more in tune if a 27-beat loop <3H, 4×2, 5×3> had been used instead of the 26-beat loop selected by the programmer <3H, 4×3, 5×2>—as Fig. 8.5 indicates: the "difference line" at the foot of the graph would be flatter if the

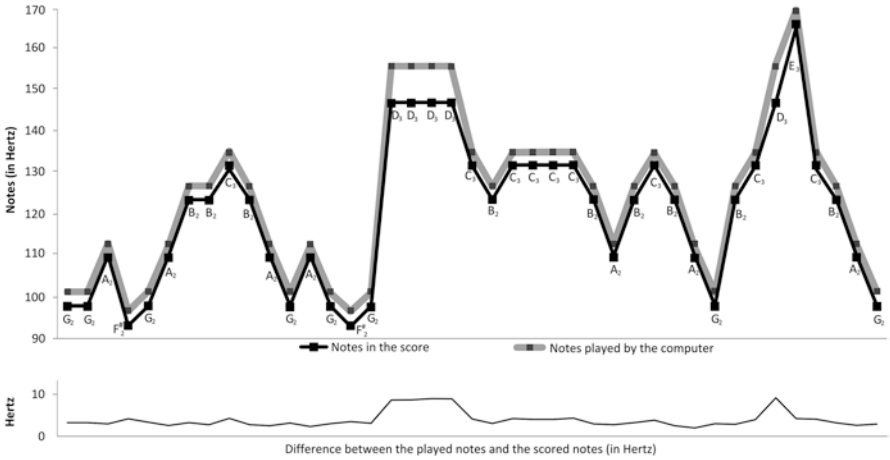


Fig. 8.5 Comparing the notes in the National Anthem's score with the notes actually played by the computer in the BBC recording. The line at the foot of the graph, the difference line, shows the displacement between the other two lines. The straighter the difference line, the more in tune with one another the notes sound. Here, the line shows that the performance sounds most out of tune in the middle and again almost at the end. In both places, the culprit is the computer's rendition of the note D_3, which the programmer pitched a little too high

27-beat loop were used. In fact, the preferable 27-beat loop even produces a frequency closer to the note's Equal Tempered frequency of 146.83 Hz (154.32 Hz as opposed to the 26-beat loop's frequency of 160.26 Hz). Why the programmer chose the excessively high 26-beat loop is puzzling.

Another puzzling example of poor tuning, this time from "In the Mood" is A_3, a note that does not feature in the other two melodies. Substituting an 18-beat loop in place of the 17-beat loop used by the programmer brings an improvement in relative tuning, and again produces a frequency closer to the note's Equal Tempered frequency of 220 Hz (231.48 Hz as opposed to 245.1 Hz in the case of the 17-beat loop). Once again, an excessively high frequency has been selected, to the detriment of the relative in-tune-ness of the notes.

8.15 Conclusion: Restoring the Recording

To put our findings to some practical archival use, we restored the British Library's recording. We increased the speed of the recording to match the original and used pitch-correction software to remove the effects of the wobble. We also filtered out extraneous noise from the recording.

Nobody had heard the true sound of the computer since the early Ferrantis were scrapped more than half a century ago. A German researcher David Link attempted to recreate the sound, by programming his emulation of the Mark II.[76] But an emulation is far from being the real thing, and without the original physical components, including of course the hooter, an emulation cannot recapture the actual sound. But now, thanks to an improbable meeting—in New Zealand—of the 1951 recording and modern analytical techniques, we really can listen to Turing's Mark II.

Our restoration is now in the British Library (reference number H3942) and can be heard at www.AlanTuring.net/historic_music_restored.mp3.

Acknowledgments With thanks to Chris Burton for assistance and advice. Copeland is grateful to the following institutions for supporting this research: University of Canterbury, New Zealand; University of Queensland, Australia; Federal Institute of Technology (ETH), Zurich, Switzerland; and Det Informationsvidenskabelige Akademi, Copenhagen University, Denmark.

References

BBC. 1950. *BBC Recording Training Manual*. London: British Broadcasting Corporation.

Bowker, G., and R. Giordano. 1993. Interview with Tom Kilburn. *IEEE Annals of the History of Computing* 15: 17–32.

[76] Link, D. 'Software Archaeology: On the Resurrection of Programs for the Mark 1, 1948–58', 2015, vimeo.com/116346967.

Campbell-Kelly, M. 1980. Programming the Mark I: Early Programming Activity at the University of Manchester. *IEEE Annals of the History of Computing* 2: 130–168.

Chadabe, J. 2001. The Electronic Century, Part III: Computers and Analog Synthesizers. *Electronic Musician.* www.emusician.com/tutorials/electronic_century3.

Copeland, B.J. 1999. The Turing-Wilkinson Lecture Series on the Automatic Computing Engine. In *Machine Intelligence*, ed. K. Furukawa, D. Michie, and S. Muggleton, vol. 15, 381–444. Oxford: Oxford University Press.

———. 2005. Introduction to 'The Turing-Wilkinson Lecture Series (1946–7)', (Turing (1946–7), in Copeland et al. (2005).

———. 2011a. The Manchester Computer: A Revised History. Part I: The Memory. *IEEE Annals of the History of Computing* 33: 4–21.

———. 2011b. The Manchester Computer: A Revised History. Part II: The Baby Machine. *IEEE Annals of the History of Computing* 33: 22–37.

———. 2012. *Turing, Pioneer of the Information Age*. New York: Oxford University Press.

Copeland, B.J. et al. 2005. *Alan Turing's Automatic Computing Engine*. New York: Oxford University Press.

———. 2006. *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*. New York: Oxford University Press.

Copeland, B.J., and G. Sommaruga 2015. The Stored-Program Universal Computer: Did Zuse Anticipate Turing and von Neumann?, in Sommaruga and Strahm (eds.) (2015).

Dean, R.T. (ed.). 2009. *The Oxford Handbook of Computer Music*. New York: Oxford University Press.

Dodd, K.N. c.1953. *The Ferranti Electronic Computer*. Armament Research Establishment report 10/53.

Doornbusch, P. 2005. *The Music of CSIRAC: Australia's First Computer Music*. Melbourne: Common Ground, 2005.

Fildes, J. 2008. 'Oldest' Computer Music Unveiled, *BBC News | Technology*, 17 June 2008, http://news.bbc.co.uk/2/hi/technology/7458479.stm.

Foy, N. 1974. The Word Games of the Night Bird (Interview with Christopher Strachey). *Computing Europe*, 15 August 1974, 10–11.

Lavington, S.H. 1980. *Early British Computers: The Story of Vintage Computers and the People Who Built Them*. Manchester: Manchester University Press.

Link, D. 2012/2013. Programming ENTER. Christopher Strachey's Draughts Program. *Resurrection* 60: 23–31.

Lukoff, H. 1979. *From Dits to Bits: A Personal History of the Electronic Computer*. Portland: Robotics Press.

McCann, D., and P. Thorne. 2000. *The Last of the First. CSIRAC: Australia's First Computer*. Melbourne: Melbourne University Press.

Prinz, D.G. 1952. *Introduction to Programming on the Manchester Electronic Digital Computer*, Moston, Manchester: Ferranti Ltd., 28 March 1952. A digital facsimile is in *The Turing Archive for the History of Computing* at www.AlanTuring.net/prinz.

Sommaruga, G., and T. Strahm (eds.). 2015. *Turing's Revolution: The Impact of His Ideas About Computability*. Basel: Birkhäuser/Springer.

Stern, N. 1979. The BINAC: A Case Study in the History of Technology. *Annals of the History of Computing* 1: 9–20.

———. 1981. *From ENIAC to UNIVAC: An Appraisal of the Eckert-Mauchly Computers*. Bedford, MA: Digital.

Tootill, G.C. 1948–9. Digital Computer—Notes on Design & Operation. National Archive for the History of Computing, University of Manchester.

Turing, A.M. 1946–7. *The Turing-Wilkinson Lecture Series (1946–7)*, in Copeland et al. (2005), pp. 464–527.

———. 1947. Lecture on the Automatic Computing Engine, in Turing and Copeland (2004), pp. 378–394.

———. 1948. Intelligent Machinery, in Turing and Copeland (2004), pp. 410–432.

———. c.1950. Turing, A. M. *Programmers' Handbook for Manchester Electronic Computer Mark II*, Computing Machine Laboratory, University of Manchester, no date, circa 1950. A digital facsimile is in *The Turing Archive for the History of Computing* at www.AlanTuring.net/programmers_handbook.

Turing, A.M., and B.J. Copeland. 2004. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life*. Oxford: Oxford University Press.

Williams, F.C., and T. Kilburn. 1952. The University of Manchester Computing Machine. In *Review of Electronic Digital Computers: Joint AIEE-IRE Computer Conference*, 57–61. New York: American Institute of Electrical Engineers.

Chapter 9
Exploring the Frontiers of Computation: Measurement Based Quantum Computers and the Mechanistic View of Computation

Armond Duwell

Abstract In this paper I investigate measurement based quantum computers (MBQCs) using the mechanistic view of computation. I begin by demonstrating that Turing's work cannot serve as an analysis of computation. I proceed to describe the mechanistic view of computation and then apply it to MBQCs. That computers as peculiar as MBQCs can be handled by the mechanistic view provides good confirmation for it. Additionally, I will show how the mechanistic view helps us understand how MBQCs work, and how this understanding might be applied to expand the frontiers of computation.

9.1 Introduction

Turing (1936/1937) sought to delimit the class of computable numbers. By "computable numbers" Turing clearly meant those numbers whose decimal expansion can be discovered by a human by following a fixed set of rules and aided by an unlimited supply of paper and pencils for calculations. In his paper, Turing developed a formalism whose elements adequately represent the essential features and processes a human would employ in calculating. What he came up with are Turing machines. Turing then delimited the class of sequences computable by Turing Machines, and given a straight-forward interpretation of those sequences, the computable numbers. So far as Turing machines are an adequate representation of human calculators, or as Turing called them, computers, he had delimited the class of numbers humans could compute. Turing (1936/1937) also showed that the Hilbert's *Entscheidungsproblem* is undecidable and sketched a proof that the class of computable numbers delimited in the paper is the same as that delimited by Church.

A. Duwell (✉)
University of Montana, Missoula, MT, USA
e-mail: armond.duwell@umontana.edu

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_9

So, what is this magic device, a Turing Machine? A Turing Machine is a type of device that consists of an infinite tape sectioned into cells, a read/write head for the tape, which can read one cell at a time and write in one cell at a time, a set of internal states and a machine table that specifies how to move the read/write head and what symbol to write based on the current internal state and symbol read from the tape, and what internal state to update to.[1] Operations can be deterministic so the movement of the tape and newly written symbol are determined by the previous state and read symbol. Alternatively, a probability distribution might describe what symbol is to be written given a state and read symbol. A Turing machine has an initial configuration and a final configuration. The initial symbols on the tape when a machine is in its starting configuration is called the input of the device. The set of symbols printed on the tape after the machine reaches its final configuration is called its output.

Turing (1936/1937) also proved that universal Turing machines exist. One can think of a machine table as a program associated with a Turing machine. It turns out that there exist machines, universal Turing machines, that have a machine table such that one can encode other machine tables into the input of the universal machine and it will simulate the action of the encoded machine table.

In a single paper, Turing resolved an outstanding problem in the foundations of mathematics, provided a rigorous analysis of what a computable number is, created computer science, and also invented the notion of a programmable computer (universal Turing machine). This is remarkable, but Turing didn't accomplish everything.

One of the things that Turing did not accomplish is to provide an analysis of computation. An analysis of computation tells us what it takes for a system to be a computing system. In computer science, at the foundational level especially, scientists often consider classes of computing systems, for example, all Turing machines, or all cellular automata, or all possible circuits. The point of an analysis of computation is not to determine the computational power of a class of computers, which, in some cases, might involve proving that they have the same computational abilities as the class of Turing machines. Instead, by identifying the essential features of computation, an analysis of computation can addresses questions like, "given that the sequence of microphysical states of a rock can be put in one to one correspondence with the sequence of states associated with a Turing machine performing a computation, is that rock actually a computing system?" Surely we have little trouble identifying paradigm instances of computers, Turing certainly gave us one; but we lack an agreed upon systematic account of underlying assumptions about computers on which computer scientists, AI researchers, psychologists, linguists, neuroscientists, anthropologists, and philosophers agree.

In this paper, I want to articulate and advocate for a particular view of computation, the mechanistic view, by applying it to the frontiers of computation. In Sect. 9.2, I will articulate the simple-mapping view of computation that exploits Turing

[1] I will use the term "state" in its normal way to correspond to all features of a Turing machine including the contents of its tape. I use the term "internal state" to describe the state of a Turing machine without consideration of the contents of its tape.

machines in the most obvious way to provide an analysis of computation and will discuss its fundamental limitations. In Sect. 9.3, I will introduce the mechanistic view of computation. In Sect. 9.4, I will introduce a very peculiar kind of quantum computer, a Measurement Based Quantum Computer (MBQC), which operates very differently than other kinds of computers. In addition to introducing the reader to MBQCs, computers at the forefront of computation, I will use them to provide strong confirmation for the mechanistic view of computation in the sense of Herschel in Sect. 9.5.

> The surest and best characteristic of a well-founded and extensive induction, however, is when verifications spring up, as it were, spontaneously, to notice, from quarters where they might be least expected, or even among instances of that very kind which were at first considered hostile to them. (Herschel 1830, 170)

MBQCs initially will seem to be difficult to analyze in terms of the mechanistic view of computation. Nonetheless I will show that they can be accommodated by the mechanistic view. Furthermore, the mechanistic view will help to highlight aspects of MBQCs that might have otherwise gone unnoticed. General lessons for computation will be drawn.

9.2 The Simple-Mapping View of Computation

Here's the basic idea behind the simple mapping view of computation.[2] Suppose that we have a paradigmatic general computing device, a Turing machine for example. It is very natural to think that if a physical system can simulate the action of a Turing machine while that Turing machine is computing something, then that physical system is a computing system. Here is one way to flesh this idea out. We assume that a Turing machine performing a computation does so in part in virtue of sequentially entering a series of different states. The simple mapping view of computation claims that if the sequence of states that a physical system evolves through can be mapped to the sequence of states that a Turing machine goes through when computing something, then it is a computing system.[3]

There are several problems with the simple mapping view of computation. The most formidable problem arises because it is too easy to match up the sequence of states that a physical system goes through to the computational states of, say, a Turing machine performing a computation. It's easy because there are just so many states that are physically accessible to systems. Macroscopic systems have on the order of 10^{23} atoms that compose them. Each of those atoms has multiple physical states themselves. So, the number of possible physical evolutions of states of macroscopic systems typically are greater than the number of possible evolutions of a

[2] For a more detailed discussion of the simple-mapping view, see Piccinini (2011).

[3] Typically, one would map equivalence classes of physical states (usually equivalence classes of microstates) to a set of computational states.

Turing machine performing a computation, so the matching can always take place. So, if one adopted the simple mapping view of computation, almost everything turns out to be a computer, thus trivializing the notion of computation that the view was supposed to provide an analysis of. This problem is known in the literature as the Putnam-Searle problem (see Putnam (1988) and Searle (1992)).

There are other difficulties for the simple mapping view of computation that are worthy of mention. We want an analysis of computation that is open to the possibility that there may be machines more efficient than Turing machines or other paradigmatic computing devices. In such cases, there may be machines that take fewer steps than Turing machines to perform a computation. In such cases, there will be no possibility of mapping the physical states of the putative computational system to the computational states of a Turing machine performing the same computational task.

One could solve this problem by extending the sequence of computational states of the device that needed fewer steps by considering extra trivial states of the device as computational. First, this is an ad hoc solution. Second, it misrepresents the computational system as requiring additional steps than is required to be computational. An analysis of computation should help us understand the computational abilities of a system, not obscure them as this solution does.

This problem points to a deeper issue. Analyses of computation should be open to the possibility that there might be computers that can possibly solve problems in very different ways than Turing machines, and still be computational. The simple mapping view seems blind to this possibility. There is a semantic issue regarding this claim that should be set aside. If one sticks with the old understanding of the word "computer", as a human with unlimited paper and pencils, then the claim that an analysis of computation needs to be open to the possibility that there are computers that solve problems very differently than Turing machines is unreasonable. I think most people understand the word "computer" differently now. They understand it as some kind of system that can do useful computational work. Most are open to the possibility that computers might be able to do things in ways that humans cannot. It is this second and broader sense of the word "computer" that I am appealing to in order to make this claim.

So, for these reasons, the simple mapping view of computation, which is the most obvious way to extend Turing's work into an analysis of computation, turns out to be a failure. In the next section, I present the mechanistic analysis of computation, which is meant to correct for the deficiencies of the simple mapping view.

9.3 The Mechanistic View of Computation

The simple mapping view of computation had many shortcomings that need to be remedied by any satisfactory analysis of computation. Towards that end, Piccinini (2007) has developed an analysis of computation that corrects for these shortcomings and more. Piccinini argues that any analysis of computation should satisfy the following desiderata:

- **Objectivity:** An account of computation should make it a matter of fact, rather than a matter of interpretation about which systems can compute a function.
- **Explanation:** It is a matter of practice that we explain the behavior of certain systems in terms of the computations they perform. E.g. we explain the behavior of some computers in terms of the programs that they execute.
- **The right things compute:** Paradigmatic computing systems must turn out to compute.
- **The wrong things do not compute:** Paradigmatic examples of non-computational systems, e.g. rocks, must turn out not to compute.
- **Miscomputation:** It is a fact that some computing systems miscompute. A miscomputation occurs when a system computes the value of a particular function, but outputs the wrong value or no value at all. An analysis of computation should make sense of how this is possible.
- **Taxonomy:** Different kinds of computing systems have different capacities. An account of computation should have the resources to explain their different capacities.

Piccinini offers his mechanistic view of computation as an analysis that satisfies the desiderata. A system performs a computation if and only if it admits a computational explanation.

A system admits of a computational explanation when it can be described as a mechanism that processes vehicles (defined as entities or variables that can change) according to rules that are sensitive to vehicle properties (Piccinini and Bahar 2013). A mechanism is a system that admits of a mechanistic explanation. A mechanistic explanation of a system X is

> …a description of X in terms of spatiotemporal components of X, their functions, and their organization, to the effect that X possesses its capacities because of how X's components and their functions are organized (Piccinini 2007, 506).

The computationally relevant components of the mechanism have the function of processing vehicles according to rules.[4] Their organization and functions give rise to the computational capacities of the system.[5] It is useful to see how this analysis applies to Turing machines. The vehicles associated with a Turing machine are finite sequences of symbols written on a machine's tape. The tape of a Turing machine is processed by the machine in accordance to a rule. The machine table associated with a Turing machine characterizes the rule that the machine operates by. Furthermore, it characterizes the functions of the components of a Turing machine.

[4] The specification of "computationally relevant" components is required because there are plenty of components in computers that are necessary, but arguably not relevant to the computations they perform, like cooling fans.

[5] It is important to note that the requirement that systems be described as special kinds of mechanisms that process vehicles according to rules is crucial. Else, any system which admitted of a mechanistic explanation would be considered computational, and there are plenty of mechanisms that we don't think are computational, e.g. refrigerators.

The organization of the components with these functions gives rise to the computational capacities of Turing machines.

Tying computation to computational explanation has several advantages. First, our intuitions about what computes and what does not are respected. Given the restrictions detailed above, the wrong kinds of things just do not compute. For example, systems like rocks don't compute because they are not mechanisms that process vehicles according to definite rules. Moreover, given the close connection to abstract computing formalisms, the kinds of systems we normally think of as being computational turn out to be computational on this analysis, e.g. circuit computers, cellular automata, register machines, etc.

Second, by making computational explanation definitive of computation, Piccinini satisfies "explanation". Furthermore, his analysis provides ample room for a robust taxonomy of computational systems in virtue of the different kind of mechanical analyses and computational explanations they have. For example, different kinds of computers can be classified in part according to the kinds of vehicles they process. Digital computers process strings of digits. Analog computers process continuous variables. Quantum circuit computers process strings of qubits.

Finally, Piccinini has placed much emphasis on the fact that his analysis can account for the phenomenon of miscomputation. Piccinini accounts for miscomputation in virtue of assigning particular functions to some components of computing systems (mechanisms). If a component fails to perform its function, a miscomputation occurs.[6]

The mechanistic view of computation straightforwardly meets all of the desiderata that have been discussed. It hasn't been discussed whether *all* of the right things turn out to compute on the mechanistic view. While all paradigmatic computing devices clearly meet the demands of the mechanistic analysis of computation, it is not clear that MBQCs will satisfy the analysis. In the next section I will introduce the reader to MBQCs and in the following section evaluate whether the mechanistic analysis of computation can handle them.

9.4 Measurement Based Quantum Computers

Measurement based quantum computers were introduced in Raussendorf and Briegel (2001, 2002). They are computers that can efficiently simulate any quantum circuit computer, which can in turn efficiently simulate any Turing machine. So, MBQCs are at least as computationally powerful as Turing machines, and certainly qualify as an important model of computation. As remarked above, MBQCs operate very differently than other computers. In order to appreciate MBQCs, and how they operate, I will not immediately describe MBQCs to the reader. MBQCs are better appreciated by first introducing circuit computers, then quantum circuit computers, and then finally moving on to MBQCs.

[6] Piccinini offers a detailed list of possible failures in his (2007).

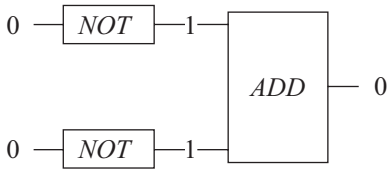


Fig. 9.1 A circuit computer for computing 1 + 1 in binary. The bits begin in the fiducial state 0. They are prepared for the computation with the NOT gates that flip the bits to the 1 state. These bits serve as inputs to the ADD gate. The output of the gate encodes the result

The classical circuit computer is composed of a set of bits, initially in some fiducial state, and a set of gates that institute dynamic changes on the bits. After all dynamic changes are instituted on the bits, the final state of the bits encodes the output of the computation. A universal set of gates is a set of gates for a circuit computer, when used in combination with one another, are sufficient to instantiate any logical function on bits.

Consider a circuit computer that would perform binary addition, more specifically addition mod 2. So, 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, and 1 + 1 = 0. Such a computer will make changes on systems that can be in one of two different states, bits. Take one of the states to represent the binary number 0, and the other to represent the binary number 1. Circuit computers use gates to make changes on the state of bits. There is a gate that does just the transitions corresponding to binary addition, call it the ADD gate (known in the literature as the XOR (exclusive OR) gate). In addition to applying the ADD gate to do the computation, one needs to consider the gates required to prepare the inputs from some initial fiducial state. So, one needs to consider, for example, the gates required to prepare one 0 state and one 1 state in order to do 0 + 1 = 1. There is a gate that does this, the NOT gate, which simply flips bit values, e.g. 0 changes to 1 and 1 changes to 0 under the operation of a NOT gate. Consider the circuit that would perform 1 + 1 and assume all bits begin in the 0 state. See Fig. 9.1.

This simple classical circuit computer obviously satisfies the requirements of the mechanistic view. Bits are the vehicles over which a rule governed process follows given the organization and function of the gates.

Quantum circuit computers are very similar to classical circuit computers, but have richer dynamical possibilities. Classical circuit computers operate on bits, which are *discrete two-state* systems. Quantum computers operate on qubits, which are *continuous two-dimensional* systems.[7] Whereas the state space of a classical circuit computer of n bits contains 2^n*states*, the state space of a quantum circuit computer is of *dimension* 2^n. A universal set of gates for a quantum computer can instantiate any set of dynamical changes to qubits to any desired degree of accuracy. In the quantum case, the Clifford group gates, CNOT, Hadamard, π/2-phase gates, and any single qubit-rotation gate *not* generated by the Clifford group gates, will be

[7] Though quantum computers operate on continuous systems, their computational power does not depend on manipulating these systems with arbitrarily high accuracy.

universal for quantum computation.[8] Similar to classical circuit computers, quantum computers can be viewed as beginning a computation in a fiducial state, all qubits in the 0 state for example, changing these qubits into the desired input state, and then performing changes on this input to produce an output state.

Quantum circuit computers fit well into the mechanical view. The vehicles manipulated by quantum computers are sequences of qubits. The function of the components of quantum circuit computers is to manipulate these qubits. They make incremental changes to qubits to produce an output vehicle, a set of qubits in the desired state, one that encodes the solution to a computational problem. The organization of the gates determines the general rule for the manipulation of vehicles that partially characterizes the computation performed by the quantum circuit. Things are not so straightforward with MBQCs.

MBQCs operate quite differently than standard circuit computers, and differently than quantum circuit computers too. MBQCs have the following parts or resources: a two dimensional array of qubits, a set of measurement devices, an information flow vector, and a classical computer. The set of qubits begins a computation in a fiducial state, but it is not a zero state. The qubits begin the computation in an entangled state. Without going into unnecessary detail, note that qubits in an entangled state are highly correlated to one another, so much so that no classical system can match those correlations. A MBQC makes changes on the qubits to perform computations, in *some* ways like a circuit computer makes changes on bits to perform computations. The way a MBQC makes changes on these qubits is by making measurements on them individually or jointly. The results of these individual measurements are random. That said, if the measurements on individual qubits are orchestrated properly, the ensuing changes on the qubits can be interpreted as a computation. See Fig. 9.2.[9]

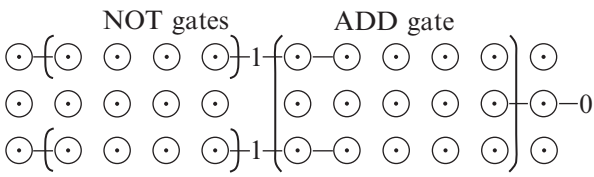


Fig. 9.2 A simplified representation of a MBQC for computing 1 + 1 in binary. The qubits, represented by circles with dots in them, begin in the fiducial state, an entangled state. Measurements are performed on the qubits in parentheses. These measurements are how one makes changes on qubits that correspond to changes made by gates in a circuit computer. The 1 s and 0 have been added as a conceptual aid. As will be described below, the state of the individual qubits associated with those 1 s and 0 does not reflect the actual state of those qubits

[8] Neilsen and Chuang (2000) is an excellent reference book for quantum computation.

[9] The figures that follow are not completely representative of MBQCs. Simplifications have been made to minimize complications, and to illustrate only the essential points. For example, the computation is depicted as corresponding to a circuit computer performing addition, but the MBQC is reversible, and the ensuing simulation of that circuit computer would be more complicated if reversibility were taken into account. See Raussendorf et al. (2003) for examples of common quantum circuit gates simulated in MBQCs.

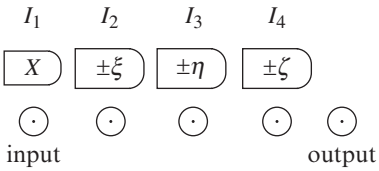


Fig. 9.3 A series of five qubits is required to instantiate an arbitrary rotation characterized by the Euler angles ξ, η, and ζ. Qubit 1 is the input qubit and qubit 5 is the output qubit. The "D" shaped boxes indicate measurements. The first is a measurement in the x direction, followed by ±ξ, ±η, and ±ζ directions, with signs determined by the results of earlier measurements. The information flow vector, **I**, is updated after each measurement, and is used to determine the appropriate sign for the next measurement via a classical computation

In order to understand how measurements are used to make changes to qubits, I want to consider an example of a gate that performs an arbitrary one-qubit rotation. The state of a quantum bit can be represented, to a certain approximation, by an arrow that can point anywhere on the surface of a sphere. An arbitrary rotation gate will simply rotate that arrow to another point on the sphere. An arbitrary rotation can be characterized by the Euler angles ξ, η, and ζ. These angles indicate the direction of rotation. A MBQC performs a rotation using a series of five qubits. Let qubit 1 be the input qubit and qubit 5 be the output qubit. Qubits 2–5 begin in a standard state, and qubit 1 is in some arbitrary state. The qubits are then entangled with one another. The input qubit is measured along the x-axis. Qubit 2 will be measured in a direction determined by ± ξ where the sign is determined by the outcome of the measurement on qubit 1. An information flow vector, **I**, keeps track of the results of measurements at each stage, and is used by a classical computer to determine the sign associated with the next measurement. Qubit 3 will be measured in a direction determined by ± η where the sign is determined by the outcome of measurement on qubit 2. Qubit 4 will be measured in a direction determined by ± ζ where the sign is determined by the outcome of measurement on qubit 3. The state of qubit 5 will then be in the desired output state.[10] See Fig. 9.3.

One of the most interesting features of MBQCs can be drawn out by considering how they simulate circuit computers (quantum or classical). Simulation of a circuit computer by a MBQC is not done gate by gate; instead it proceeds in what might be described as a holistic manner. An arbitrary gate in a circuit can be approximated to any degree of accuracy by combinations of the measurement patterns associated with the one-qubit rotation gate mentioned above, and four other measurement patterns. These other measurement patterns correspond to the Clifford group gates mentioned above. It turns out that these additional measurement patterns do *not* depend on the results of earlier measurements, unlike the measurement pattern for a qubit rotation. In fact, the measurements associated with these patterns can all be performed *simultaneously* and *initially* in the computation. It should be noted that

[10] Technically, the final state will differ from the desired state by a set of rotations (I, σ_x, σ_y, or σ_z, the Pauli matricies and the identity). The information flow-vector mentioned above keeps track of these rotations to ensure the proper computational result.

this includes the qubits that correspond to the inputs and outputs in the quantum circuit simulated by a MBQC! One can then proceed to make measurements that depend on the results of all of those initial measurements, and then those that depend on them, and so on.

The earlier example of addition can be revisited to emphasize this strange feature. Measurements on certain sets of qubits corresponded to the operation of gates in a circuit computer for computing 1 + 1. One of the most interesting features of MBQCs is that parts of the measurement patterns associated with gates on a circuit computer, namely those that do not depend on the results of any other measurements, can be performed simultaneously, and initially. So, *part* of the changes on qubits that correspond to preparation of inputs *as well as* addition itself can be performed simultaneously, and initially. Additionally, the qubits corresponding to the outputs of a quantum circuit computer can be measured initially. What this means is that the changes that a MBQC makes do not mimic the temporal order of changes of a circuit computer.

An additional oddity of MBQCs is that the results of the individual measurements performed on the qubits are random, and so too is the evolution of the information flow vector up until its final state. That said, the random outcomes of the measurements are nonetheless exquisitely correlated with one another. The correlations generated by entanglement are what allow one to wring out this randomness and extract useful computational results from MBQCs. See Fig. 9.4.

A different way to make the point is this. The information flow vector changes randomly. That said, if one updated the information flow vector by flipping a coin suitably many times instead of making measurements on quantum systems and let

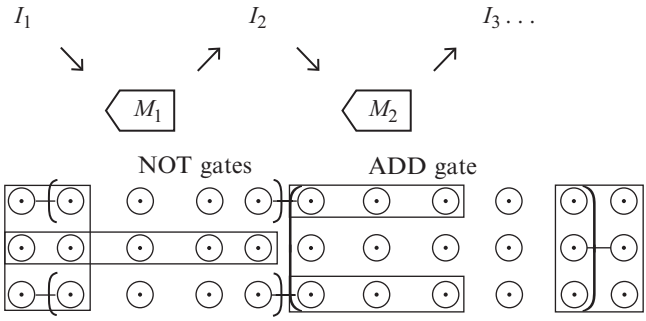


Fig. 9.4 A MBQC for computing 1 + 1 in binary. The qubits, represented by circles with dots in them, begin in the fiducial state, an entangled state. Measurements on qubits in *parentheses* correspond to changes that gates make in circuit computers. Qubits that are *boxed* together represent qubits that can be measured simultaneously and initially. Note the overlap between boxed qubits and qubits in parentheses. This represents components of transformations performed by gates that can all be performed simultaneously. The M-tabs correspond to the set of measurements that are performed simultaneously, and which are generally determined up to a sign by the information flow vector, denoted by **I**, which is updated after each round of measurements. For the sake of simplicity the boxed qubits do not represent the actual qubits that would be measured simultaneously were one to simulate 1 + 1 = 0, but the conceptual point is the same as if they were

the rest of the computation proceed, it would fail miserably. Though the information vector changes randomly after each measurement, entanglement ensures that these individual random changes are correlated in just the right way to do useful computational work. In the next section, we discuss whether the mechanistic view of computation can accommodate these exotic computers.

9.5 MBQCs and the Mechanistic View

Recall that that the mechanistic view requires that computing systems be functional mechanisms that process vehicles according to definite rules. The task is to see how well one can fit MBQCs into this framework. It is easy enough to see that MBQCs have components that are organized in a particular way. That said, to see components as processing vehicles, and to assign definite functions to these components, is more difficult. In this section I will describe the difficulties in fitting MBQCs into the mechanistic view and show how they can be resolved. I will also show that application of the mechanistic view to MBQCs can revise our understanding of how one can compute the value of a function at a particular point.

MBQCs utilize classical computers whose vehicles are strings of digits. They also utilize quantum systems. Quantum circuit computers are well-described as processing strings of qubits. As has been described above, qubits do not flow through MBQCs as they do in quantum circuit computers. The correlations between qubits in the computational cluster due to entanglement make the flow of qubits in the system unnecessary for the performance of computational work. So, MBQCs are a hybrid computing system that utilizes two kinds of vehicles, strings of digits, and the entire cluster of qubits.

What is interesting about the two different vehicles in MBQCs is that they have an influence on one another. Measurements performed on qubits alter the state of the cluster of qubits as well as generate a string of digits, the information flow vector that encodes the results of those measurements. The string of digits then gets processed by a classical computer that determines the angles up to sign for the next round of measurements on the qubit cluster. This is how the classical vehicles alter the quantum vehicle.

It is clear that functions can be assigned to components of a MBQC that process strings of digits. The function of the components is to determine measurement angles based on the results of earlier measurements, and in the final stage to decode the solution to the computational problem from the results of the final round of measurements on the qubits. Assigning functions to the components that make measurements on the qubit cluster is more difficult.

It is difficult to view the components that perform measurements on qubits as having the function of processing vehicles. It is difficult to view them as having the function of processing strings of digits because the digits generated by the measurement devices via the results of measurement are probabilistically independent of the strings of digits entering the devices (determining the appropriate signs for mea-

surement angles). Similarly, it is difficult to assign measurement devices the function of processing quantum vehicles. The digits entering the devices determine the sign of the measurement angles, but the resultant state of the qubits measured is probabilistically independent of these signs. Furthermore, once qubits are measured, they are never interacted with again in the course of the computation. So, it is difficult to assign measurement devices a function relating either to the processing of a classical vehicle, or a quantum vehicle. It is difficult, but not impossible.

The way to assign a function to the measurement devices to avoid the mistake of thinking of a cluster of qubits like one would a string of digits. When one alters a single digit on a string, one does not alter the state of the rest of the digits that compose the string. The same is not true of qubits. In the case of entangled qubits, making measurements on some subset of these qubits alters the state of these qubits, but also the state of the qubits that were *not* measured.[11] Einstein called this spooky action at a distance. This spooky action at a distance is essential to assigning the measurement devices a function. They alter the state of the non-measured qubits in a way that drives the state of the cluster closer to one that encodes the result of the computational problem. That is their function. So, MBQCs do fit well into the mechanistic view, despite appearances to the contrary.

Given that MBQCs can fit into the mechanistic view, I want to show what insights they shed on computation generally from within the mechanistic view. The mechanistic view of computation characterizes computers as processing vehicles. It is often very natural to view an early vehicle in the computation as representing an input associated with the computational task being performed and being sequentially processed. One particular case is when a computer evaluates a function, and the input corresponds to the point of the function the computer is to evaluate. In the case of digital computers, Piccinini (2007) endorses this view explicitly. Digital computers have strings of digits that enter a system, persist within that system, are manipulated by some components of the system according to definite rules, and finally exit that system. Quantum circuits computers work in a similar way. MBQCs do not quite work that way.

As was mentioned above, MBQCs have considerable flexibility in when certain measurements take place. In particular, when simulating a circuit computer any Clifford group gates that would be performed in a circuit computer can be performed simultaneously and initially, even though they generally correspond to dynamic interactions with inputs, outputs, and everything in between. In some cases, dynamic interactions with inputs can even be delayed until later in the computation too.[12] Though dynamic interactions of a MBQC correspond to the set of

[11] For those familiar with interpretations of quantum mechanics, one might wonder whether this is true in all interpretations. It is straight-forwardly true in dynamic collapse theories and pilot-wave theories. It is true in the Everettian interpretation true, at least as formulated by Wallace (2012). The measurements cause the emergence of worlds that alters the state of non-measured qubits.

[12] For example, suppose that a MBQC is simulating a circuit computer. In the case where in the circuit computer that is simulated a qubit only undergoes changes corresponding to the Clifford group gates, the corresponding changes in a MBQC can be delayed until the end, even those changes that correspond to preparation of inputs. Aaronson and Gottesman (2004) have showed

dynamic changes instituted by a circuit computer, *the temporal order of these changes does not*. So, generally, in MBQCs, one cannot straight-forwardly view computation of a value of a function as having an input corresponding to the point to be evaluated enter into the computer, persist, be manipulated, and finally exit the machine.

Though the temporal ordering of dynamic interactions of a MBQC do not always closely parallel the temporal ordering of dynamic interactions in a circuit computer that they might simulate, there is something that compensates for this. Just like in the case of circuits, the spatial arrangement of devices that institute dynamic changes play a crucial role in MBQCs. So, even though the temporal order of measurements corresponding to gates in a circuit computer do not match the temporal order of the action of those gates, the spatial arrangement of the measurements made on the qubit cluster does.

The general point is this: the computational task that a system performs might not be easily determined by tracking the vehicles that that system processes. Our intuitions that if a computational system evaluates a function at a point, that a vehicle corresponding to that point will enter the system and be sequentially processed to alter the vehicle to the output, are misleading at best, and mistaken at worst.

9.6 Conclusion

Turing made remarkable contributions to computer science, and he certainly inspired many analyses of computation, though he himself didn't provide one. In this paper I sought to demonstrate the limitations of using Turing machines as a basis for an analysis of computation. I offered the mechanistic view as a superior analysis of computation and applied it to a recent advancement in computation, the discovery of MBQCs. I have argued that MBQCs provide ample confirmation to the mechanistic view of computation.

I have shown how applying the mechanistic view helps us understand what is interesting and different about MBQCs. They demonstrate that a single computational system might employ multiple kinds of vehicles. They also teach us that computer components don't necessarily wear their functions on their sleeves. Functions of components might only be determinable when considering the components of the system as a whole and their relations. Furthermore, MBQCs teach us that the sequential processing of vehicles might not correspond in a natural and obvious way to the points of a function evaluated by a computer. These features are in striking contrast to some of the features of Turing machines.

To conclude, let me mention one way in which these general insights drawn from application of the mechanistic view to MBQCs might be applied. Computationalism,

---

that the Clifford group gates alone can be implemented to solve problems in the complexity class Parity-L. So this is the class of problems for which MBQCs have complete temporal flexibility with respect to when dynamic changes are instituted on qubits.

the view that cognition is explained by neural computations, has been the dominant theory of cognition for the last six decades. That said, the nature of neural computation has been hotly contested. Some claim that neural computation is digital computation, some claim that it is analogue, and some claim it is neither, but that it is still computational nonetheless (Piccinini and Baher 2013). To date there is no widely agreed upon positive characterization of neural computations. The application of the mechanistic view draws attention to useful possibilities that might help us understand neural computation. Is the nervous system a hybrid computer utilizing digital, analogue and/or other vehicles? Might the functions computed by nervous systems be obscured in similar ways to those of MBQCs? I don't know the answer to these questions, but I hope to have demonstrated the conceptual value of applying the mechanistic view of computation to help us explore and expand the frontiers of computation.

References

Aaronson, S., and D. Gottesman. 2004. Improved Simulation of Stabilizer Circuits. *Physical Review A* 70: 052328.

Copeland, B.J. 1996. What Is a Computation? *Synthese* 108 (3): 335–359.

Herschel, J.F.W. 1830. *Preliminary Discourse on the Study of Natural Philosophy*. New York: Johnson Reprint Corporation.

Neilsen, M.A., and I.L. Chuang. 2000. *Quantum Computation and Quantum Information*. Cambridge, UK: Cambridge University Press.

Piccinini, G. 2007. Computing Mechanisms. *Philosophy of Science* 74: 501–526.

———. 2012. Computation in Physical Systems. In *The Stanford Encyclopedia of Philosophy* (Fall 2012 Edition), ed. Edward N. Zalta. http://plato.stanford.edu/archives/fall2012/entries/computation-physicalsystems/.

Piccinini, G., and S. Bayar. 2013. Neural Computation and the Computational Theory of Cognition. *Cognitive Science* 34: 453–488.

Putnam, H. 1988. *Representation and Reality*. Cambridge, MA: MIT Press.

Raussendorf, R., and H.J. Briegel. 2001. A One-Way Quantum Computer. *Physical Review Letters* 86: 5188.

———. 2002. Computational Model Underlying the One-Way Quantum Computer. *Quantum Information and Computation* 6: 443.

Raussendorf, R., D.E. Browne, and H.J. Briegel. 2003. Measurement-Based Quantum Computation with Cluster States. *Physical Review A* 68: 022312.

Searle, J. 1992. *The Rediscovery of the Mind*. Cambridge, MA: MIT Press.

Turing, A.M. (1936/1937). On Computable Numbers, with an Application to the *Entscheidungsproblem. Proceedings of the London Mathematical Society* 42(2): 230–265.

Wallace, D. 2012. *The Emergent Multiverse: Quantum Theory According to the Everett Interpretation*. New York: Oxford University Press.

Chapter 10
Embodying Computation at Higher Types

S. Barry Cooper

Abstract Since Turing's early speculations, progress with the building of intelligent machines has been slow, with obstacles tending to be dealt with in ad hoc rather than theoretical ways. This view of history is reflected in Marvin Minsky's comment (Boston University, 2003) that "AI has been brain-dead since the 1970s".

The optimism of Turing's reported "I am building a brain" (Hanslope Park, 1944) has withered before the reality of the chimera that is embodied intelligence. There is a growing sense that the universal Turing machine has given rise to a paradigm (McCulloch and Pitts (1943), Kuhn (1996)) which, as well as being powerful and successful, is misleading in important respects: Firstly, the disembodiment implicit in the Turing machine was unreal even for the early construction of a digital computer, and this has become more apparent with the modern respect given to natural computing in its various guises. And more abstractly, this has made us more aware of the extent to which theory has failed to give due prominence to the role and structure of information in the computational processes that science seeks to characterize.

The theme of this article is the extent to which embodiment, information, and its type structure are relevant to the material and mental world, and their computational processes. In particular, it is argued that 'computation' in the real world is more general—and consequently harder to control—than is usually imagined.

10.1 The Mathematician's Bias

In October 2010 the Association for Computing Machinery, "the world's largest educational and scientific computing society", launched an online Symposium entitled "What Is Computation?". The Symposium was hosted by the web-based magazine Ubiquity under the umbrella of the Communications of the ACM. *Ubiquity* editor-in-chief Peter Denning's introduction to the Symposium explained (2010):

> What is computation? This has always been the most fundamental question of our field. In the 1930s, as the field was starting, the answer was that computation was the action of

S.B. Cooper (deceased)
School of Mathematics, University of Leeds, Leeds LS2 9JT, UK

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_10

> people who operated calculator machines. By the late 1940s, the answer was that computation was steps carried out by automated computers to produce definite outputs. That definition did very well: it remained the standard for nearly fifty years. But it is now being challenged. People in many fields have accepted that computational thinking is a way of approaching science and engineering. The Internet is full of servers that provide nonstop computation endlessly. Researchers in biology and physics have claimed the discovery of natural computational processes that have nothing to do with computers. How must our definition evolve to answer the challenges of brains computing, algorithms never terminating by design, computation as a natural occurrence, and computation without computers?

Lance Fortnow would have probably represented a majority of readers when he responded in December (2010):

> Alan Turing, in his seminal 1936/37 paper "On computable numbers, with an application to the *Entscheidungsproblem*", directly answers this question by describing the now classic Turing machine model. The Church-Turing thesis is simply stated:
>
> Everything computable is computable by a Turing machine.
>
> The Church-Turing thesis has stood the test of time, capturing computation models Turing could not have conceived of, including digital computation, probabilistic, parallel and quantum computers and the Internet. The thesis has become accepted doctrine in computer science and the ACM has named its highest honor after Turing. Many now view computation as a fundamental part of nature, like atoms or the integers.
>
> So why are we having a series now asking a question that was settled in the 1930s?
>
> A few computer scientists nevertheless try to argue that the thesis fails to capture some aspects of computation. Some of these have been published in prestigious venues such as Science … , the Communications of the ACM … and now as a whole series of papers in ACM Ubiquity. Some people outside of computer science might think that there is a serious debate about the nature of computation. There isn't.

Before the 1990s, public dissent from this view would have been a bold step, risking ridicule and damage to academic standing. Things have changed, and the ACM Symposium is indeed a sign that the ground is moving under our feet. The revolution in progress has yet to reach Copernican scale, though promising a change in our relationship with causality—with our very conception of causality—as far-reaching as any previous shift in world-view. A key contribution to the Symposium was Dennis Frailey's "Computation Is Process" (2010)—including the words:

> The concept of computation is arguably the most dramatic advance in mathematical thinking of the past century. … Church, Gödel, and Turing defined it in terms of mathematical functions … They were inclined to the view that only the algorithmic functions constituted computation. I'll call this the "mathematician's bias" because I believe it limits our thinking and prevent us from fully appreciating the power of computation.

The view we give voice to below—increasingly widely held—is that the discussion is about more than the nature of computation. It is part of a growing appreciation of the fact that information is not flat, and that the accompanying computational modeling must be able to cope with the underlying causal confusions widely affecting science and the humanities. Mathematicians, even those aware of the wider challenges, have largely kept to their technical and conceptual comfort zones. No wonder there is widespread disillusionment regarding the role of mathematics in this informationally extended world. For Nassim Taleb, with his "ludic fallacy" (2007), what mathematicians can deal with is close to his *definition* of what we

should not be interested in. Those that deal with the mathematics of information beyond the reach of computation, and of language even, are largely invisible.

10.2 Causality Under Scrutiny

The ability to mentally image, process, and give meaning to emergent form in nature involves a computational process which the human mind manages quite impressively. The below termite "cathedral" is produced by relatively primitive organisms which one might simulate on one's computer with some success, including a degree of interactivity: but neither termite nor computer will simulate the appreciation of emergent form, of higher order information, of which the human brain is observed to be capable. Still less will the termite represent that higher order information in a form suitable for accessing by further computational routines (Smolensky (1988)). We will need to look more carefully at the causal character of emergence in the light of a closer engagement with the underlying information, via various contexts (Fig. 10.1).

The mathematical character of the challenge presented by causality is specially clear from the physics. As Lee Smolin elaborates on at length in *The Trouble with Physics: The Rise of String Theory* (2006, p. 241) "causality itself is fundamental". Causality is key to higher order concepts, while being itself in need of clarification. For instance, regarding the tendency to relate questions about determinism to ones concerning causality, John Earman says (1986, p. 5):

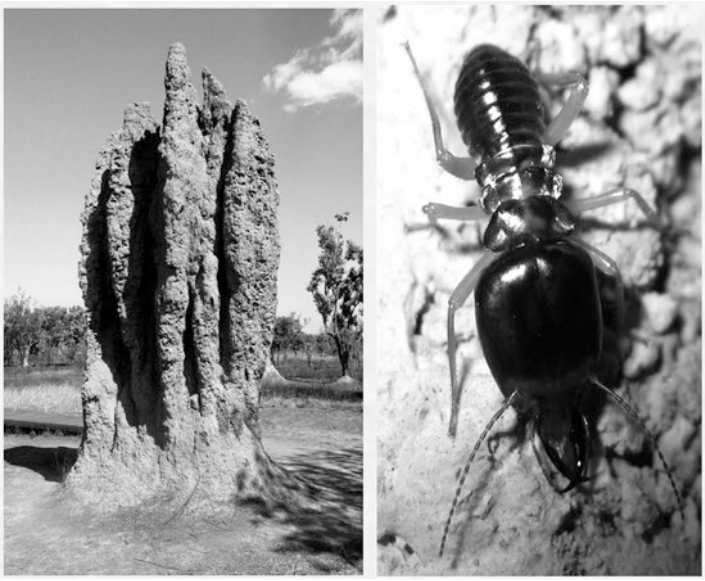


Fig. 10.1 Termite-Cathedral (Available under creative commons from http://en.wikipedia.org/wiki/Emergence#mediaviewer/File:Termite_Cathedral_DSC0357.jpg)

> … the most venerable of all the philosophical definitions [of determinism] holds that the world is deterministic just in case every event has a cause. The most immediate objection to this approach is that it seeks to explain a vague concept—determinism—in terms of a truly obscure one—causation.

From our perspective, part of the confusion arises from the attempt to regard causality as being a simpler concept than that of determinism. While one might be happy with higher order entities being determined, the concept of causality is generally attached to more local phenomena—events, component individuals. According to the contemporary focus on process, emergence, chaos, turbulence, connectionist models of computation and global features of social and economic contexts, it is this that leads to philosophical problems with causality, and a parallel dissatisfaction with the scope of the classical model of computation. Here is Albert Einstein raising a particularly fundamental issue about the standard model of physics—one which is still with us. Implicitly the complaint is that, according to what we know, causality in physics is incomplete (1969, p. 63):

> … I would like to state a theorem which at present can not be based upon anything more than upon a faith in the simplicity, i.e. intelligibility, of nature … nature is so constituted that it is possible logically to lay down such strongly determined laws that within these laws only rationally completely determined constants occur (not constants, therefore, whose numerical value could be changed without destroying the theory) …

Over the years Einstein's concern has been echoed by many others (Weinberg (2002) and Zeil (2013)). For instance Peter Woit, in his book *Not Even Wrong—The Failure of String Theory and the Continuing Challenge to Unify the Laws of Physics* (2006), says:

> One way of thinking about what is unsatisfactory about the standard model is that it leaves seventeen non-trivial numbers still to be explained, ….

In recent months, CERN watchers have looked in vain for confirmation of supersymmetry, an important route to clarifying a number of issues—such as dark matter—muddying our view of the universe. Looking for causality even further up the informational scale, no less than Alan Guth, author of *The Inflationary Universe—The Quest for a New Theory of Cosmic Origins* (1997), asks about the determination of natural laws:

> If the creation of the universe can be described as a quantum process, we would be left with one deep mystery of existence: What is it that determined the laws of physics?

If our incomplete grasp of the causal character of the physical universe is unsatisfactory, even more so is the 'causality abrogated' that the uncertainty phenomenon in quantum mechanics pushes us towards (Penrose (1987)).Taking its cue from the probability-bound collapse of the wave function consequent on a measurement, and Hugh Everett III's many-worlds setting for entanglement, we now have a byzantine landscape of multiverse proposals, structured by Max Tegmark into his Multiverse Levels I-IV (2003). The abrogation finds its quintessential expression in the anthropic principle, delivering our universe of conveniently appropriate fundamental constants and natural laws by some sort of cosmic accident. David Deutsch in his

1997 *The Fabric of Reality* (1997) sees the existence of quantum computers as pointing to the existence of the multiverse. Scott Aaronson, in *Quantum Computing Since Democritus*, comments (2013, p.149):

> … a quantum computer is *not* a device that could "try every possible solution in parallel" and then instantly pick the correct one. If we insist on seeing things in terms of parallel universes, then those universes all have to "collaborate"—more than that, have to *meld into one another*—to create an interference pattern that will lead to the correct answer being observed with high probability.

10.3 Causality Fragmented, Supervenient … Simulated?

So causality is basic—but a 'truly obscure concept', incomplete, abrogated—and, unavoidably fragmented in the context of the human brain. There are many examples of philosophers (particularly) rejecting Cartesian dualism in uncompromising style—echoing Susan Blackmore's "grand illusion of consciousness". Here is a quote from Professor Blackmore, the analysis in terms receptive to an information-theoretic interpretation (1999, p. 220):

> Dualism is tempting but false. For a start no such separate [thinking] stuff can be found. If it could be found it would become part of the physical world and so not be a separate stuff at all. On the other hand if it cannot, in principle, be found by any physical measures then it is impossible to see how it could do its job of controlling the brain. How would immaterial mind and material body interact? Like Descartes' "thinking stuff", souls, spirits and other self-like entities seem powerless to do what is demanded of them.

Such issues are visited in depth by the philosopher Jaegwon Kim, echoing such questions as: How can mentality have a computational role in a world that is fundamentally physical? And what about "overdetermination", the problem of phenomena having both mental and physical causes? In his *Physicalism, or Something Near Enough* (2005) Kim puts it:

> … the problem of mental causation is solvable only if mentality is physically reducible; however, phenomenal consciousness resists physical reduction, putting its causal efficacy in peril.

The persistent problems in reconciling mentality with its physical host are arranged by philosophers who care about such things under the heading of supervenience (Kim 1998, pp. 14–15):

> … represents the idea that mentality is at bottom physically based, and that there is no free-floating mentality unanchored in the physical nature of objects and events in which it is manifested.

Put more mathematically (McLaughlin and Bennett 2014):

> A set of properties A supervenes upon another set B just in case no two things can differ with respect to A-properties without also differing with respect to their B-properties.

The widespread consensus around supervenience hosts a huge variety of ideas regarding exactly how mentality is anchored in the physical. Interestingly, even a

fairly committed reductionist like Daniel Dennett favors an analysis with different orders of information, even if it does not directly relate to the supervening (cf. Dennett (2013)). The term 'subconscious' may be a misleading one—with consciousness supervening on the physical, but unconscious mental activity—even that associated with reflex sensorimotor activity—being mediated by a developing consciousness in a quite nonlinear and selective fashion (Leijonhufvud (2010)). In his *Consciousness Explained* (1991) Dennett can be found plausibly referring to unconscious "thoughts" as a higher order phenomena (p. 308):

> Unconscious thoughts are, for instance, unconscious perceptual events, or episodic activations of beliefs, that occur naturally—that must occur—in the course of normal behavior control. Suppose you tip over your coffee cup on your desk. In a flash, you jump up from the chair, narrowly avoiding the coffee that drips over the edge. You were not conscious of thinking that the desk top would not absorb the coffee, or that coffee, a liquid obeying the law of gravity, would spill over the edge, but such unconscious thoughts must have occurred—for had the cup contained table salt, or the desk being covered with a towel, you would not have leaped up. Of all your beliefs—about coffee, about democracy, about baseball, about the price of tea in China—these and a few others were immediately relevant to your circumstances. If we were to cite them in an explanation of why you leaped up, they must have been momentarily accessed or activated or in some way tapped for a contribution to your behavior, but of course this happened unconsciously.

There are more observable signs of fragmented causality at work: with the task being to provide the wherewithal of a successful partnership between different levels. For Alan Turing back in the 1940s it was a steep learning curve. As Solomon Feferman describes this in his magical (1988, pp. 131–2):

> Turing, as is well known, had a mechanistic conception of mind, and that conviction led him to have faith in the possibility of machines exhibiting intelligent behavior.

Since that time, scaling the gap between computer and mentality has presented investigators with a challenging phase-transition—an obstacle theoretically based in the 'causality' of the underlying informational structure, we will argue. For those working at the practical level, the theoretical approach—via the logic and mathematical modeling—has been singularly unrewarding. For Rodney Brooks, the experimental robotic route has made more sense. For him (2001):

> … neither AI nor Alife has produced artifacts that could be confused with a living organism for more than an instant.

On the other hand, such ad hoc investigations leave artificial intelligence theoretician and pioneer Marvin Minsky unimpressed—commenting in characteristically trenchant terms at Boston University in May 2003: "AI has been brain-dead since the 1970s".

In his final years, Turing was already anticipating difficulties ahead, with thinking molded by a focus on the human dimension. For instance, the mistake-prone young writer of the seminal "computable numbers" article comes to mind when we read, in the final paragraph (1947):

> … if a machine is expected to be infallible, it cannot also be intelligent. There are several theorems which say almost exactly that.

And in Turing's popular piece on "Solvable and Unsolvable Problems" (1954) we read in his final paragraph (p. 23):

> The results which have been described in this article are mainly of a negative character, setting certain bounds to what we can hope to achieve purely by reasoning. These, and some other results of mathematical logic may be regarded as going some way towards a demonstration, within mathematics itself, of the inadequacy of 'reason' unsupported by common sense.

The human brain itself uncannily reflects elements of the Brooks-Minsky dialectic, and of the Turing preoccupation with cooperation between machine and human. In his (2009, p. 174) *The Master and his Emissary: The Divided Brain and the Making of the Western World*, Iain McGilchrist describes how:

> The world of the left hemisphere, dependent on denotative language and abstraction, yields clarity and power to manipulate things that are known, fixed, static, isolated, decontextualised, explicit, disembodied, general in nature, but ultimately lifeless. The right hemisphere by contrast, yields a world of individual, changing, evolving, interconnected, implicit, incarnate, living beings within the context of the lived world, but in the nature of things never fully graspable, always imperfectly known—and to this world it exists in a relationship of care. The knowledge that is mediated by the left hemisphere is knowledge within a closed system. It has the advantage of perfection, but such perfection is bought ultimately at the price of emptiness, of self-reference. It can mediate knowledge only in terms of a mechanical rearrangement of other things already known. It can never really 'break out' to know anything new, because its knowledge is of its own representations only. Where the thing itself is present to the right hemisphere, it is only "re-presented" by the left hemisphere, now become an *idea* of a thing. Where the right hemisphere is conscious of the Other, whatever it may be, the left hemisphere's consciousness is of itself.

Why should there be different kinds of thinking, and why should the brain architecture so separate their physical hosts (Damasio (1999))? Is this a purely ad hoc, even accidental, adjustment within the evolutionary process, or is there something more fundamental at work? The enduring role of natural selection, in the context of the growth of complexity theory, leads us to expect something 'more fundamental'. As Steven Pinker attempts to persuade us, with a degree of success (1997, pp. 161–162):

> The "complexity" that so impresses biologists is not just any old order or stability. Organisms are not just cohesive blobs or pretty spirals or orderly grids. They are machines, and their "complexity" is functional, adaptive design: complexity in the service of accomplishing some interesting outcome. … No set of equations applicable to everything from galaxies to Bosnia can explain why teeth are found in the mouth rather than in the ear. And since organisms are collections of digestive tracts, eyes, and other systems organized to attain goals, general laws of complex systems will not suffice. Matter simply does not have an innate tendency to organize itself into broccoli, wombats, and ladybugs.

There is certainly an observed level of robustness of the brain architecture across a very wide spectrum of living organisms. Though there will be doubts about the degree to which the 'design' is well enough defined, in terms of mechanism, to be properly termed 'design'.

A key aspect of the brain architecture of placental mammals, such as humans, is the corpus callosum, connecting and mediating the functionality of the separate hemispheres. McGilchrist comments (2009, pp. 18–19):

> … the evidence is that the primary effect of callosal transmission is to produce *functional* inhibition.
>
> … it turns out that the evolution both of brain size and of hemisphere asymmetry went hand in hand with a reduction in interhemispheric connectivity. And, in the ultimate case of the modern human brain, its twin hemispheres have been characterized as two *autonomous* systems.
>
> So is there actually some purpose in the division of neuronal, and therefore, mental processes? If so, what could that be?

We might further ask: Given this division, to be found in animals generally, what is the benefit of the moderated reconnection via the corpus callosum? And what is the explanation of the placental mammal connection? Relevant to our more general concerns with the computational content of causality in a very general context, we might reframe these questions:

What are the computational parallels and distinctions between the different modes of dealing with information? And between the ways in which the computational modes can effectively combine? How do these modes relate to the classical Turing model of computation? And is there a suitable adaptation of classical computability theory capable of accommodating them in an informative way? And crucially for life in our computationally hybrid world—what is the appropriate balance, and level of interaction and autonomy, to be granted these computationally different frameworks? And what are the mechanisms, or absence of mechanisms, to ensure such a balance? (Fig. 10.2)

The hope is that we are seeing a new level of exchange of experiences and analytical tools with the potential to give us not just enhanced human functionality, but

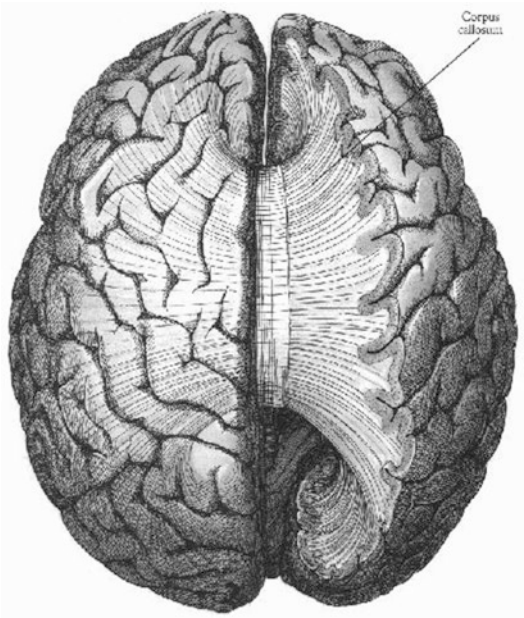


Fig. 10.2 Corpus callosum (Reproduced from David Hubel, "Eye, Brain and Vision", available at http://hubel.med.harvard.edu/book/b34.htm

a new consciousness of the special relationship between different ways of thinking—based in a suitably updated and more mature appreciation of the nature of causality and its computational framework.

A first step is to look more closely at the classical model.

10.4 Causality as Computation—"I am building a brain"

Despite the distrust of those with a more spiritual, artistic or humanistic view of their world, the scientific world we inherited from Isaac Newton and his contemporaries has served us well. Einstein's comment (1950, p. 54):

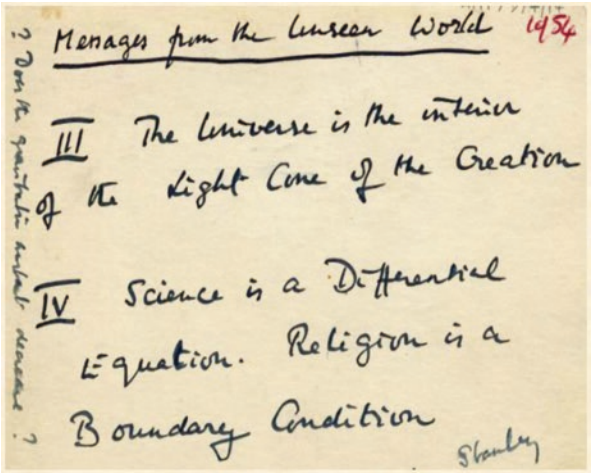
> When we say that we understand a group of natural phenomena, we mean that we have found a constructive theory which embraces them …

underpins a hard-nosed approach to knowledge of the world, which expects epistemology to provide off-the-peg knowledge for mass consumption, knowledge machine reproducible via proofs and computer calculation. The mechanics of the mathematical model underlying so much of the science was originally extracted from a well-defined physical context. Turing based them on the actions of human "computers" (usually women in pre-1936 days) calculating in a given language, with pencil and paper, according to specific instructions.

The close relationship of the model to the physical world reflected Max Newman's description of Turing as "… at heart more of an applied than a pure mathematician"—although the model became relevant in many other contexts: To computing machines, physics, and biology, for instance. Turing had a lifelong interest in basic elements of science and engineering. But this was not part of what was depicted in the (1936/7) paper on computable numbers. As a schoolboy, Turing had studied Einstein's theory of relativity; and in Cambridge had attended lectures of Paul Dirac on quantum mechanics. Robin Gandy, Turing's only doctoral student, actually started out as an applied mathematician, and was employed as such as a lecturer at Leeds University in the late 1950s. The postcards Turing sent to Gandy before he died in 1954 (Fig. 10.3) seem to us now to show the influence of Dirac and the keen interest in questions of causality and computation raised by the new science.

The Turing machine disembodied a rather specific part of the real world, and served to disembody our vision of computation much more widely—and, arguably, held in tow a deal more disembodiment than was appropriate.

Having erased the human 'computers' from the picture, the disembodied hardware appeared trivial. It consisted of no more than an extendable tape subdivided into cells in which to record symbols, and a single reading/writing head—see Fig. 10.4. While the basic actions performed by the reading head were chosen to generate all the more complicated actions one might envisage, ensuring that the potential was there to compute any real number computable by an algorithm. Essentially, with the right usage, the Turing machine hardware was 'Turing complete', in that (modulo some less than trivial re-embodiment) it could compute anything the


Messages from the Unseen World 1954

III The Universe is the interior of the Light Cone of the Creation

IV Science is a Differential Equation. Religion is a Boundary Condition

Stanley

? Does the quantitative instinct decease ?

Fig. 10.3 Turing to Robin Gandy, 1954 (Courtesy of the turing digital archive, AMT/D/4)

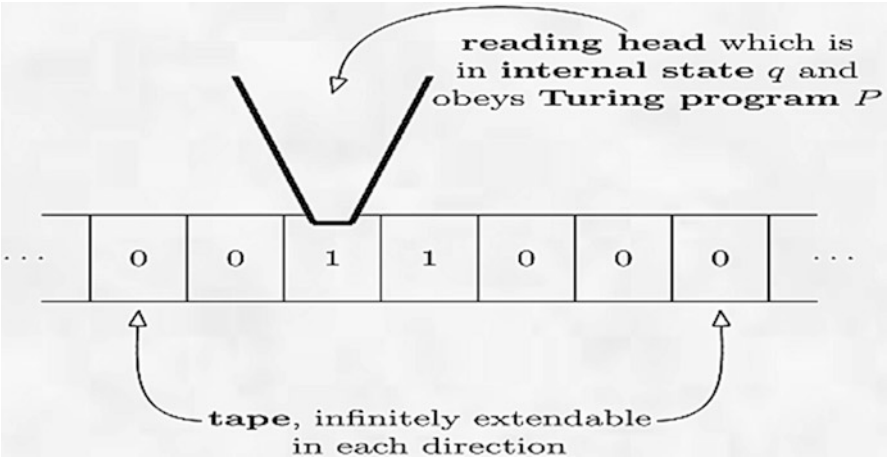


Fig. 10.4 A Turing machine (Reproduced from Cooper 2004)

human 'computer' could compute. And various people duly verified that all the other abstract models of such computation which they encountered—some apparently quite different—had exactly the same power. Of course, different models could be applied to different languages, with various computing conventions, but this was immaterial in relation to overall computing potential (Fig. 10.4).

So the hardware was as simple as could be, and all the computing power was in the program it was presented with. And—something which underlies the sometimes fractious contacts between theoretical computer scientists and engineers— the convention arose whereby the theorists identified the machine with its program. Having

disembodied the computing machines of earlier centuries—the machines which had to be physically built pre-Turing to do different kinds of computational chores—the theorists muddied the waters by giving the program, something which was not a machine at all, the name "Turing machine". In the course of time, the distinctions between machine and information, and between program and data have receded in relation to both the engineering and theoretical worlds. And the seeds of this were laid by Turing's very special *universal machine*, which he used to derive a computational counterpart to Kurt Gödel's 1931 Incompleteness Theorem.

To understand the significance and mathematical sleight of hand encased in the universal Turing machine, one needs a little more detail than is usually given. The important property the universal machine (that is, its program) had to have, was crucially: That all one had to do was to give the universal machine a piece of data which said "Please, compute with input x just like that other Turing machine over there (which you never saw before) would", and it would do it! The trick was, that instead of a lumbering great contraption, like Charles Babbage's Analytical Engine would have been if built, which would be hard to code up as a single piece of input data(!)—a Turing machine was entirely determined by a finite set of instructions, a set which held within it all the logical structure needed for a particular computational chore: it could hence be coded up as a single piece of input data—say a number, or a binary string, or whatever the machine was used to.

So let's assume we have coded up all the Turing machines as natural numbers e, say, via a listing of all the Turing machine programs. We can then program our universal Turing machine **U**, say, so that when it is given a pair (e,x) of natural numbers, it decodes the machine **T** coded by e, and then proceeds to compute exactly like **T** would with input x. You can actually write down the program for such a U, and this has been done in various forms. We write φ **T** (φe) for the number theoretic function of one argument computed by **T**.

The huge significance of the coding trick is that we have reduced some complicated information (a suitably set up Babbage machine, say) to a simple piece of data. You might not be too impressed. You might say "Ah, but all I need is a set of plans of the machine, and I can build it—and then a code for the plans is essentially the machine—what's the big deal?" The "big deal" is that you are using Turing's work with hindsight. If I gave you a bucket of slime mold (see, e.g., Adamatzky et al. (2013)), the coding capturing the computing medium might be a little trickier … i.e., non-trivial, and just perhaps an indication of a more complicated level of information that our Turing machine cannot handle. Later, we will look at the type structure of information, and look on the reduction of machine to code by Turing as a route to less simple computational type reduction. And following Turing one must then view the machine itself, regardless of the informational type a description it demands, as an input into the computational process.

In the more limited context of computer history, it needs to be noticed that the actual computing machines built involved a high degree of re-embodiment. It was not just the discovery of basic elements of the Turing machine hardware, such as material hosts for the memory, which were required. For very important operational

reasons, basic logical structure was embodied in the hardware. In the more iconoclastic modern re-assessment on disembodied computation—which this article is intended as a contribution to—various writers (usually from outside mathematics) have immersed themselves in the minutiae of particular computer solutions, and discovered that Turing's universal machine was fairly unrecognizable therein (see for example (Haigh 2013)). Though from a mathematical perspective, the big idea of universality can be swarmed all over with Lilliputian thoroughness, its powerful vitality is hard to keep down.

It is the universal machine, and its strangely fortuitous genesis in type reduction, that underpins the functionalist view of the informational universe and its computational structure—and then brings it into doubt as a comprehensive computing paradigm. The functionalist view—which stresses what a computer does as something realizable via *different computing platforms*—was first developed in relation to philosophy of mind and AI. A seminal contribution was Hilary Putnam's writing on "Minds and Machines" (1960). The idea can also be found in muted form in the notion of *virtual machine*, emerging in the computing world in the late 1960s, for instance in the IBM time-sharing operating system CP/CMS (Control Program/Cambridge Monitor System).

The post-1936 developments were dramatic, and must have been very exciting for those in at the start of the computer age. One can excuse Turing for getting carried away, at one point quoted by Donald Bayley as saying, while working on the 'Delilah' speech encryption project at Hanslope Park, that he was "building a brain" (1983/2012, p. 304).

In recent times, Putnam has drawn back somewhat from the 1960s functionalist simplicity, retaining some elements and disposing of others (1967/1975, 1975, 1988a, b, 1990). Here he is (1984, p. 89 1990 reprint):

> … if the physical universe itself is an automaton … , then it is unintelligible how any particular structure can be singled out as "the" causal structure of the universe. Of course, the universe fulfills structural descriptions—in some way or other it fulfills every structural description that does not call for too high a cardinality on the part of the system being modeled … the problem is how any one structure can be singled out as "the" structure of the system.

This does not derail the power of the paradigm. There are many successful reductions of "natural" examples of computation to the Turing model. A particularly striking example is David Deutsch's (1985) use of the quantum Turing machine model to bring the 'standard model' of quantum computation within the computationally classical fold. But there is a giant step from this to Deutsch's assertion (2006) that:

> I am sure we will have [conscious computers], I expect they will be purely classical, and I expect that it will be a long time in the future.

10.5 Complexity, Emergence, and 'Levels of Abstraction'

Turing's last published paper is not only his most cited, but also the most dramatically prescient following the (1936/37) career-launching "On computable numbers". Today, at a time when computational models from nature and bioinformatics frame the professional lives of literally thousands of scientists, it is instructive to revisit the computational underpinnings of Turing's patterns in nature. In 1936 the young Alan Turing had broken a computational context into basic ingredients which possessed a clear computationally causal character. In his (1952) paper on the chemical basis of morphogenesis Turing attempts something similar for a selection of pattern forming processes in nature. The differential equations he formulated from the proposed reaction-diffusion models gave solutions suggesting that the analysis was broadly correct. Research since 1952 has partly confirmed Turing's approach, and partly led to modifications and new directions. What is important for us is that Turing had effectively given descriptions of the morphogenesis in computational terms, which pointed the way to notions of computation, embodied computation, with the potential to bring together the descriptive and computational approaches to epistemology. It opened out a route to a better understanding of "how the world computes". For more explanation and history of Turing's work and influence in this area, good sources can be found in Cooper and Leeuven (eds.) (2013); particularly informative are the articles by James Murray (2013) and Philip Maini (2013).

The solutions to Turing's differential equations were point-wise computable, and Turing was able to simulate them in Manchester using the new computer technology—in effect, placing these surprising outcomes firmly within the classical framework. However, the descriptive form of the solutions had the power, potentially, to resist this kind of—what we will call—*type reduction*. Notice that what is interesting about the simulations—and morphogenesis in nature—are the patterns, whereas the computer's focus is point-wise. It occupies the world of the termite, computing without being able to take full ownership of its creation. Turing's vision brings us closer, not just to understanding morphogenesis as a phenomenon, but to a better understanding of levels of computation and how they are achieved (Fig. 10.5; cf. Cooper (2011)).

Concerning the problem of pattern recognition and extraction of meaning, Webb (1980, pp. 246–7) writes:

> Our discussion of abstract Turing machines has, of course, taken the problem of 'pattern recognition' largely for granted, but it is by no means obvious that a physical realization of a Turing machine can always be programmed to learn to recognize the global properties of arbitrarily degraded symbols. … Weizenbaum … points out that the general problem of computer vision "is in many respects fundamentally the same as that of machine understanding of natural language" …. The historical roots of this problem go back … to the issue of blind mechanisms doing geometry that arose in Poincaré's critique of Hilbert's "purely logical" approach to geometry. … I believe that the ultimate test of artificial intelligence—at least in the case of mathematics—will come in this geometric arena which

requires the coordinated use of both the perceptual and cognitive abilities of machines, as opposed to the relatively barren arithmetical arena in which machines need only think.

Poincaré's (1921) observation turns out to be a key one, backed up by subsequent developments.

Pattern recognition is a first step towards extracting semantic content from patterns. Let us illustrate this with an example from mathematics, where both computational content and the human appreciation of higher type information is familiar to us. The Mandelbrot set provides a simple example of a computationally based structure which carries with it higher order information of great interest to many leading researchers—see for example, Blum et al. (1997). The fascinating form of the Mandelbrot set emerges via an approximation to its representation in the complex plane, embodied on the observer's computer monitor. Despite the book of Blum et al. (1997), there are questions concerning the point-wise computability of the set. The members of the set are the complex numbers c for which the limiting behaviour of the sequence $z_0 = 0, z_1, z_2, \ldots$, governed by the recurrence relation

$$z_{n+1} = z_n^{\ 2} + c$$

is bounded.

The logician will easily spot that the description of such numbers c involves the addition of a couple of alternating quantifiers, a source of alarm as we shall see that quantification was what takes the observed behavior of Turing's universal machine**U** out of the purview of itself. The mathematics tells us enough about the bound involved to reduce the description to one universal quantifier, giving the description of the Mandelbrot set the flavor of Turing's (1936/37) example of an *incomputable* set.

For the core computable analysis community the computability of the Mandelbrot set is still correspondingly in question. Interesting as this is, it is not what impresses us out in the real world. Our mental image of the form of the set is unrestricted by the point-wise computations, and we delve deeper and deeper via our computer approximations, excavating surprise upon surprise, aesthetic pleasure upon aesthetic pleasure, as the infinite variety of shape within is unfolded. It is not the facility of the brain for receiving and internally representing the shapes which is so remarkable—it is the sense of context which comes into play, the resource-based expectations, the higher order comparison with previously stored form and the mental repercussions which feed into our subsequent thinking. Echoing Judson Webb, it is "by no means obvious" that our computer can be programmed to share this experience, still less to iterate the shared experience as the human brain might (Fig. 10.6).

The problem here parallels the challenge of "big data" in today's wider world. In this context great strides are being taken. An appreciation of the mathematical context of the statistical tools investigators are driven to here may enable us to better understand the potential convergence of real-world models, its scheduling, and the future relationship of the available models—human and human constructed.

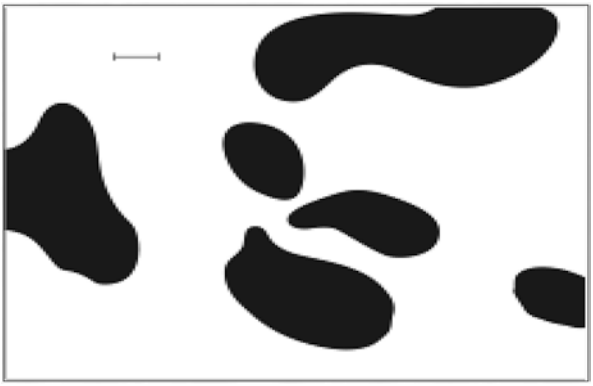


Fig. 10.5 Morphogenesis: Dappled pattern on a cow's back (Reproduced from Turing 1952)

Underlying this scenario (see Sect. 10.6 below) is the role of descriptions, their relationship to Turing's classical computation, and the way in which this relationship creates informational structure relevant to our residence in the real world. There are different entrypoints to this landscape. There is the "levelism" of the rich repertoire of mathematical hierarchies, mainly the domain of the logician—in general, mathematicians like to be given a specific informational context to solve their puzzles within. And there is the descriptive natural language based approach of the mass of humanity, at its most analytical reaches represented by the philosophically minded—and trained. Of course, there are increasingly people who can traverse with a degree of expertise the boundaries between different approaches and disciplines (see, e.g., Abramsky (2013)).

A recent, and timely, intervention from the philosophical side is Luciano Floridi's book on *The Philosophy of Information* (Oxford University Press, 2011). Levels of Abstraction play a key role in his advocacy of ISR (Informational Structural Realism) as an alternative to the informational and computational flatness of the dominant digital ontology. According to Floridi (2011):

> A level of abstraction (LoA) is a finite but non-empty set of observables. … The introduction of LoAs is often an important step prior to mathematical modelling of the phenomenon under consideration … Use of LoAs is effective … where a typed theory would be effective … [but] analysis … may be conducted at different levels of epistemological levelism.

Floridi is keen to emphasize the full diversity of LoAs, without being too specific about their basis. The more appropriate mathematical analysis is to a given context, such as in physics or other physical sciences, the more prepared he is to admit the relevance of informational typing. Of course, the mathematical analysis needs to respond to the way in which the "levelism" can be transcended by relationships and reductions between levels—duly captured conceptually, if not in specific detail, by corresponding aspects of the mathematics; while the semantics need to recognize relationships between the constructive levers to hierarchical levels of information,

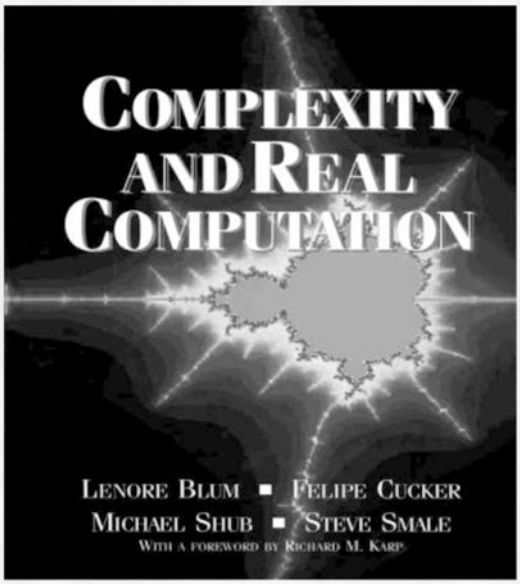


Fig. 10.6 A leading textbook in higher-order information theory

and the physical processes and phase transitions which impact on the nature and frontiers of the domains identifiable semantically as LoAs.

This impact is as significant in the ontological and epistemological domains as it is in the more abstract logical and mathematical contexts. The typing arose from Bertrand Russell's work on setting rules for the epistemology which purged the science of paradoxical description of sets. Kurt Gödel gave a simple description of the basic typing in his "Russell's Mathematical Logic" (1944, p. 135n):

> By the theory of simple types I mean the doctrine which says that the objects of thought ... are divided into types, namely: individuals, properties of individuals, relations between individuals, properties of such relations, etc. … , and that sentences of the form: "*a* has the property φ", "*b* bears the relation *R* to *c*", etc. are meaningless, if *a*, *b*, *c*, *R*, φ are not of types fitting together. Mixed types (such as classes containing individuals and classes as elements) and therefore also transfinite types (such as the class of all classes of finite types) are excluded. That the theory of simple types suffices for avoiding also the epistemological paradoxes is shown by a closer analysis of these.

We can clearly see the typing at work in the transition from individual type-1 members of the Mandelbrot set to its conflated type-2 whole. And the commonly encountered descriptive progression from underlying computational causality to emergent form can be more generally related to the type-theoretic framework.

The improved clarity concerning LoAs and their relationships pays off. It makes better sense of the way in which descriptions leading to new scientific (epistemological) levels substantiate and are clarified by what we know about the computational structure accompanying the typing, and the finer mathematical infrastructure carried with it. We need to carry this forward. Without it we are only a step ahead of

the termites in our understanding of the emergent forms and the observed LoAs. What is emergence, and why is it difficult to identify when it is taking us beyond the classical model? As Ronald C. Arkin points out (1998, p.105):

> Emergence is often invoked in an almost mystical sense regarding the capabilities of behavior-based systems. Emergent behavior implies a holistic capability where the sum is considerably greater than its parts. It is true that what occurs in a behavior-based system is often a surprise to the system's designer, but does the surprise come because of a shortcoming of the analysis of the constituent behavioral building blocks and their coordination, or because of something else?

Appearances without analysis can be misleading, as the so-called "British Emergentists" of the inter-war years discovered (see McLaughlin (1992), pp. 49–93). This despite the fact that the conceptual grasp was impressively modern, anticipating the today's widespread use of the notion of emergence, in very different contexts. Philosopher Charlie Broad, at Cambridge at the time as Turing, characteristically wrote (Broad (1923), p. 623):

> ... the mental properties of those events which do have mental properties are completely determined by the material properties which these events also have ... it is certainly not ... a form of *Reductive* Materialism; it is a form of the theory … of *Emergent* Materialism.

10.6 Higher Type *Incomputable* Information & Randomness

Turing's interest in morphogenesis, and the discovery of descriptions of emergent patterns in nature based in *computationally characterized* causality, gave us a better understanding of how another level of information could arise—*computed* (in some sense to be clarified) by nature—with new properties of its own. Emergence takes us to a new world of higher type information computed in a higher type fashion. And in the new world of higher type computation, we must get used to things being rather different, and paradigm breaking. We must face challenges to universality, closed computational context, and an extended Church-Turing thesis.

On the positive side, we may build a bridge from the digital ontology of the scientist and classical computational model, to the world of language, common sense, imagination and intuition, still incorrigibly inhabited by most of us: a world which haunted Alan Turing throughout his life, as we observe in his "popular" writings and broadcasts after the war, and even glimpse in occasional informal comments in his more technically arcane writings, such as in his (1939) "ordinal logic" paper. One can think of this bridge between intellectual heuristics as a form of virtual corpus callosum: a kind of abstract correlate of the complex connectivity between two kinds of thinking, a connectivity special to placental mammals, and most highly developed in humans.

Of course, the main aim of Turing's (1936/37) paper was not just to model basic computational practice, but to also show that the model was mathematically transcended. *Incomputability* was the *point* of the paper—not the laying of the theoretical foundations of computer science. And in a mathematically precise way he

mapped out the steps up a level of the type structure, a Jack and the Beanstalk scaling of the heights leading to a new world of dangers and opportunities. The ascent would be easier than the way back, a familiar part of the computational terrain. But with ingenuity one might contend successfully with the 'big data' and deliver—like Churchill's workers at Bletchley Park—the golden egg of decrypted information content.

Taking a universal Turing machine **U** and a natural number x, and one can ask whether there is a computer program to tell us (in general) whether **U** given input x ever stops computing. In (1936/37) Turing (essentially) proved the unsolvability of this so-called Halting Problem for **U**:

> No computer can tell us, for each given input x, whether **U** will ever complete its computation with input x.

Actually, what Turing did was have his machines compute binary real numbers. And a computable real $r \in (0,1)$ would be one whose non-terminating decimal representation was computed by a Turing machine **T**, say, writing $r = r_{\mathbf{T}}$. Then Turing gave us an effective version of Cantor's theorem concerning the uncountability of the open interval (0,1). In the spirit of Gödel's Incompleteness Theorem, Turing showed that **U** could not have computational knowledge of its own collated outputs (the jump in type): there were Turing machines **T** for which **U** could not tell whether the decimal representation $r_{\mathbf{T}}$ was a fully computed real or not. Otherwise, one could computably list all computable reals, computably diagonalize the list, and get a new real $r_{\mathbf{T}*}$, say, not on the list.

The more usual incomputable object associated with **U** is its Halting Set

$$\mathrm{H}_{\mathbf{U}} = \{x \in \mathbb{N} \mid \varphi_{\mathbf{U}}(x) \text{ is computed}\}.$$

The power of first-order Peano Arithmetic (PA) to 'semi-represent' the Halting Set gave undecidability of PA—and most strikingly, the negative solution to David Hilbert's (1928) *Entscheidungsproblem*, in the form of "Church's Theorem" saying, in simple terms:

> No computer can tell us, for each given sentence of first-order logic, whether it is logically valid or not.

Church's Theorem can seem somewhat counterintuitive at a first encounter, though our frustrating experiences with embodied computers prepares us for the unpredictability of computational outcomes!

Out in the real world, unpredictability is omnipresent, without shaking the confidence of many in the basic validity of the classical model, and its comprehensive extendability. To others, it seems more likely that we can blame faulty epistemology for an imperfect understanding of the structure of embodied information than for the unpredictability of the phenomena it encloses. Just as we had to get used to the reality of wave-particle duality at the quantum level, we may yet have to accept the crushingly simple observation that both quantifiers and limits provide instruments for the ascending of the informational type structure. And that machine, as embod-

ied higher type information will not always come with parallel devices (statistical or otherwise) for type reduction.

Given that the observer will meet difficulties in observing beyond her operative LoA, particularly in traversing phase transitions between basic causal context and emergent form encasing it (in either direction), it will not be so easy to validate speculations concerning the computational character of these informational frontiers. Some of the most theoretically intractable problems in science and the humanities can be located around such boundaries—between quantum and classical reality, between brain and mentality, and around the balance between the human individual and the emergent social and economic outcomes we seem unable to control.

We conclude this section with a few remarks on how the incomputability accords with what we observe in the outside world, in particular, in regard to the tendency to replace the robustly defined notion of incomputability in the real context with the more arbitrary notion of randomness (cf. Downey and Hirschfeldt (2010)), Mardia and Cooper (2012)).

If the precise character of probabilities governing the outcome of a quantum measurement are determined by higher order rules emergent from subatomic structure, what is this subatomic structure? We know from work with networks and emergent biological form that the particularities are often key to the global outcome. The invoking of an absolute randomness to relieve us of the task is philosophically unsatisfying. The mathematician knows that there is no such thing as a robustly defined absolute randomness. There are different approaches to clarifying randomness as a meaningful concept, with some degree of convergence at the very basic levels. But randomness is about avoidance of obstacles to our intuitive vision of the concept. And it is in the pinning down of the obstacles that we discover that there is a whole inexhaustible hierarchy (even hierarchies) of randomness.

Of course, quantum randomness is a familiar experimental and theoretical phenomenon for the physicist. As computer scientist Cristian Calude says: "It passes all reasonable statistical properties of randomness."

So the question arises: Just how random is quantum randomness? And given that mere incomputability can look like randomness—maybe quantum randomness is not random at all? Such questions are hard to get a grip on. In a joint paper on Quantum Randomness and Value Indefiniteness ((2008), pp. 165–168), Calude and physicist Karl Svozil formulate some basic assumptions concerning the quantum physics, and on the basis of these manage to demonstrate the incomputability, but not as yet any proof of any level of randomness.

The economics presents very different problems. There are certainly basic computational mechanisms at work in the economic context. But the economics is subject to forces which originate in diverse and quite different contexts, in particular, within the minds of individuals. There is a corresponding interplay of the global and local, with the human aptitude for absorbing and representing observed macroeconomic activity for further economic intervention, presenting the economist with complexities crying out for the identification of emergent pattern. This identification must needs be based on observation, as must any resulting computational structure imposed on what one takes as significantly emergent. If one attends conferences

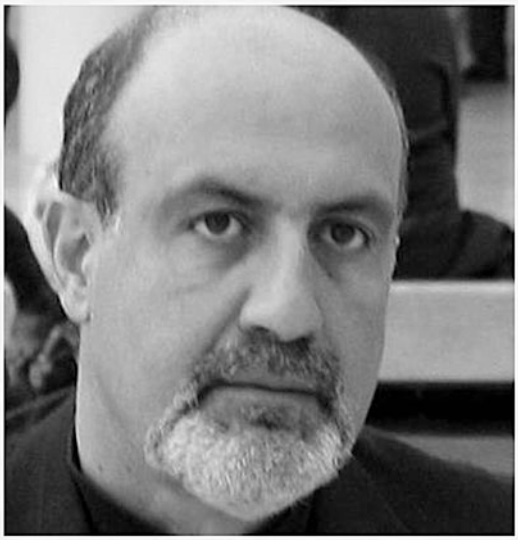


Fig. 10.7 Nasim Taleb

of economists, one finds different approaches to pinning down the patterns, sometimes taking the cue from more theoretical areas such as physics, with econophysics one of the more interesting and active areas. But here is Nassim Taleb (2007) again, decrying the all-too-often failure of the computational and mathematical approaches:

> I have spent my entire life studying randomness, practicing randomness, hating randomness. The more that time passes, the worse things seem to me, the more scared I get, the more disgusted I am with Mother Nature … The more I think about my subject, the more I see evidence that the world we have in our minds is different from the one playing outside.

Again, in what sense is it randomness? Is it incomputability? Or even some complexity-theoretic toy version of classical incomputability? If one wants to distinguish between paradigms, one needs to know a lot about both the reality and the theory one is seeking to impose on it (Fig. 10.7).

K. Vela Velupillai is one of the select few economists who have a good understanding of the logic, and in particular, the computability theory. His view of the economic uncertainties that Taleb points to, by which we are all beset, is nicely captured in his (2005 (cf. Velupillai et.al. (2011)):

> Through [parametric Diophantine equations], we enter the weird and wonderful world of undecidabilities and uncomputabilities, which is why the economic world, in its macroeconomic and microeconomic incarnations, is full of financial ambiguities. To pretend that the economic world can exist without such ambiguities and, hence, occasional and systematic exploitation of logical loopholes by unethical and immoral practitioners of financial wizardry is dangerous. On the other hand, recognising the intrinsic inevitability of such ambiguities may mean that we might happily, in a quantitative economics of the future, return to the Linnean fold, to classify and systematise particular intractable accounting schemes in increasingly and morally transparent ways.

It is in Velupillai's spirit of seeking out the source of the balance between unpredictability and form in computationally complex environments that we focus more closely on the mathematics underlying the apparent randomness.

Before doing that, it is worth saying that computational complexity does not always arrive in the material world red in tooth and claw. Georg Kreisel (1970), an early thinker on stronger versions of the Church-Turing thesis, speculates on the potential for incomputability in the apparent simplicities of Newtonian dynamics, via the underlying infinitary mathematics, proposing a collision problem related to the 3-body problem, which might result in "an analog computation of a non-recursive function".

10.7 Meaning What We Say—and Computing What We Mean

When Ludwig Wittgenstein says (1921, Proposition 1), "The world is everything that is the case", he is setting out on a path consistent with longstanding intuitions concerning the role of language and symbolic representation. It is an intuition which predates the scientific era, one dignifying our representations of the world via language and picture, granting them meaning relevant to how things really are. Having developed this view, his concluding main Proposition 7 (1921): "Whereof one cannot speak, thereof one must be silent" brings with it a granting of the world an epistemological, maybe even ontological, elusiveness evidenced by the failure of our representations—of the rule of "facts".

In *Alan Turing—His Work and Impact* (Cooper and van Leeuven (eds.) (2013)), in her commentary Juliet Floyd points to important parallels in Turing and Wittgenstein's respect for and understanding of the function of language (Floyd 2013, p. 250):

> Wittgenstein and Turing are often regarded, in a misleading caricature, as philosophical opponents. Wittgenstein is taken to be a humanistic philosopher of meaning and "forms of life", hostile to mathematical logic and the very idea of a Turing machine; Turing is taken to be a mechanistic or behaviouristic theorist of the mind, intent on reducing the concept of meaning to that of information. Neither picture is correct …
>
> Wittgenstein and Turing shared … a particular sort of anti-reductionist attitude toward logical and conceptual analysis. On their view, it is the everyday, purposeful uses we humans make of language that crucially animate and frame the notions of meaning and information.

What Alan Turing had done in (1952) for the emergence of form in nature is to capture particular examples via descriptions using the limiting mechanisms of the differential and integral calculus. He had also shown in (1936/37) how relatively modest extensions of the language used to describe Turing machines and their computations might lead (for example, via the Halting Set for a universal Turing machine), to levels of incomputability captured for us via the descriptions.

The intuition is that natural phenomena not only generate descriptions, but arise and derive from them. The potential for reconciling on the one hand the pre-Newtonian hybrid heuristic yielding a world captured informally via symbolic representation, with—on the other hand—the more modern scientific vision limited by modeling under controlled computational conditions, is striking. Out of such a reconciliation emerges a more coherent world, bringing together the historic experience of the power of descriptions for building a picture of reality, with an appropriate analysis of a logically parallel language and its computability theoretic infrastructure.

In mathematics, the experience of very real and novel features of structures arising via descriptions is a familiar one. And classically, one has no problem accepting such structural features as important and interesting characteristics, even if the logical complexity of the descriptions is beyond that acceptable within the computational domain. The formal notion, with which few outside of logic are very familiar, is that of *mathematical definability*. And yes, as well as the real world of ad hoc definitions, one which is important in mathematics and beyond, the logicians have developed a framework within one can discuss what, out there in the real world, one might hope to describe—or what is theoretically beyond description. There is also a working out of relationships between computation and description! This could be useful if more people knew what the logicians, a very few of them working in this area, were up to (Rogers (1967), Soare (1987)).

The 1930s work of the logician Alfred Tarski in formalizing a notion of definability of truth in a structure was a seminal development, helping build the framework, not just for a better understanding of basic algebraic and number theoretic structures, but for the better understanding via hierarchies of the relationship between definability and computability theoretic structure (see Tarski (1931)).

Hans Reichenbach's *Axiomatik der relativistischen Raum-Zeit-Lehre* (1924), on axiomatizing special relativity, was an early indication that twentieth century science would test post-Newtonian strictures defining what a good scientific theory should deliver, bringing *definability* in science back into focus. The axiomatization of general relativity is a current project, carried forward by the Budapest group of Istvan Nemeti and Hajnal Andreka.

For the ontological realism one might find a mathematical approach via invariance under comprehensive enough automorphisms more appropriate. There is much discussion of multiverses and alternative realities in relation to the physics. To properly constrain the possible alternatives one does need to filter in a comprehensive underlying causal structure, and a proper mathematical analysis of the possible representations in terms of an analysis of the potential automorphism groups. Anyone who has worked with this kind of mathematical challenge across a range of structures will know that 'many worlds' are nowhere as easy to manipulate into existence as one might expect. But more on this later. The basic question is how one models the basic underlying causal structure. For invariance a partial modeling may suffice. For failure of invariance, for ontological ambiguity, such as that encountered at the quantum level, one needs to be comprehensive. Note, that the sort of determination of natural characteristics—laws, constants, the geometry of space-time—that come

to mind in this context, is unlikely to be algorithmic, or mechanical in any recognizable sense. And this challenges us to model the higher order mode of natural 'computation' via suitably generalized computability theoretic notions. The aim is to provide computational approaches to this higher world of real information, a mathematics with properties well enough devised and understood for us to substantiate its link with what we observe.

Of course, a definition in terms of basic causality can be viewed as a corresponding computation, and descriptions grounded in a logic appropriate to the context will deliver invariance of what they define. In this sense, a definition pinning down an invariant feature in nature does persuasively 'compute' that feature. We have seen this in terms of emergent form in nature, where the termite cathedral we saw earlier is there to be appreciated, an embodied example of a type-2 computation.

But unlike the Turing patterns, we do not have a description of the termite cathedral. Turing found his descriptions by paying attention to the specific chemical iterations underlying the pattern formation. And this is not an uncommon phenomenon in the real world. One often needs to take a more explicitly computational route, carefully adapted to the observed process, to provide a computational framework with an explanatory and efficiently computational role. And it is this which the various notions of higher finite type are designed to tackle. For a history and review of notions, see Longley (2005). Early work grew out of the classical case, with key contributors being Stephen Kleene and Gerald Sacks (Fig. 10.8).

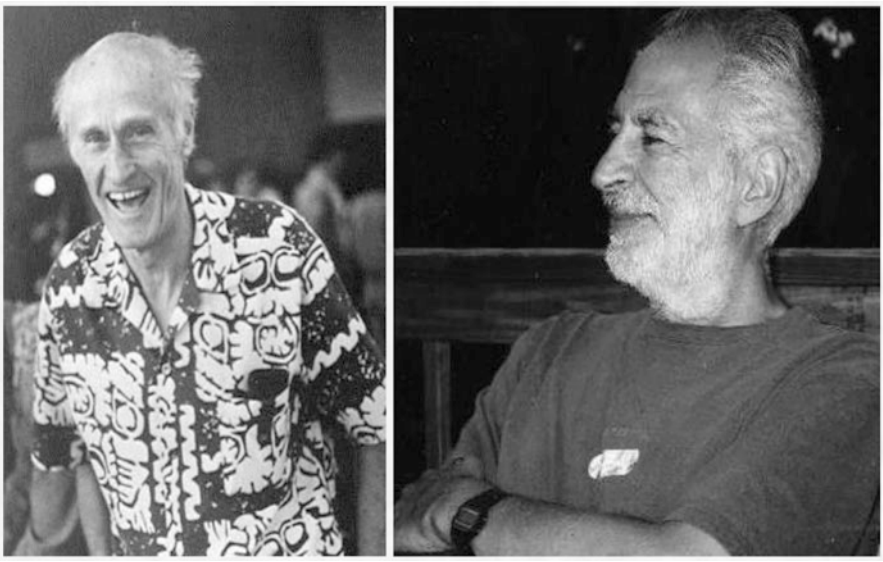


Fig. 10.8 Stephen Kleene (*left*) and Gerald Sacks (*right*) (Reproduced from Halmos 1987 (http://www-history.mcs.st-and.ac.uk/PictDisplay/Kleene.html) and http://www.math.harvard.edu/~sacks/)

As Longley describes though, the notions have multiplied, and the computational frameworks confirms much of what we observe regarding the uncertainties attendant on the bringing of the mathematics of turbulence, emergence or big data into our classical computational comfort zone. Regarding the conceptual robustness we have come to expect, Longley writes (2005):

> It is … clear that very many approaches to defining higher type computability are possible, but it is not obvious a priori whether some approaches are more sensible than others, or which approaches lead to equivalent notions of computability. In short, it is unclear in advance whether at higher types there is really just one natural notion of computability (as in ordinary recursion theory), or several, or no really natural notions at all.

The definitive book on the subject promises to be Longley and Normann's book.

10.8 A Computationally Host for Definability

Our underlying theme has been the extent to which mathematics can give substance and informative structure to emergent intuitions about how the world works. The background has been Alan Turing's corpus callosal joining together of shoe and star gazing. Above we commented on how Turing's discovery (along with Alonzo Church) of incomputability connected with Nassim Taleb's insights concerning economics and "randomness" (2007). In his (1939) paper on "Systems of Logic Based on Ordinals" we see how Turing grapples with incompleteness and incomputability, using computable ordinals during the deconstruction of mathematical phase transitions, while delivering observations on intuition and ingenuity—echoing by those of Jacques Hadamard (1945), based on Henri Poincaré's *Société de Psychologie* lectures on creative thinking and free will (1921). Alan Turing's (1952) examples of mathematical descriptions of morphogenic emergence anticipated later developments across a whole spectrum of disciplines, and gave us a key insight into the role of definability in structuring information, while helping us understand its elusive computational character.

There exists a mathematical basis for the contemporary content of these anticipations of the role of definitions, definability and their bearings on what we can compute. This is informed by the buzz of current interest in big data, emergence and the familiar fuzzy thinking, based in statistical approximation, that makes us human. We keep in mind John Longley's questioning of the likelihood of us being able to establish a Church-Turing thesis above the type-one level.

Douglas Hofstadter and Emmanuel Sander have written a whole book (2013) about definability—craftily concealed within a detailed focus on the role analogies play in our thinking. They refer (2013, p.184) to "the recognition of a correspondence between a newly-minted mental structure and an older one—in short, the construction of an analogical bridge". In emphasizing—what we recognize as—a computational difference between an analogy and a categorization, they describe:

> —the seemingly trivial case of the recognition of a cup as a cup. Suppose you are at a friend's house and want to fix yourself a cup of tea. You go into the kitchen, open a couple of cup-

> boards, and at some point you think, "Aha, here's a cup." Have you just made an analogy? If, like most people, you're inclined to answer, "Obviously not—this was a *categorization*, not an analogy!", we would understand the intuition, but we would propose another point of view. Indeed, there is an equally compelling "analogy" scenario, in which you would have just constructed inside your head a mental entity that represents the object seen in your friend's cupboard. In this scenario, you would have created a mental link between that mental representation and a pre-existing mental structure in your head—namely, your concept named "cup". In short, you would have created a bridge linking two mental entities inside your head.

What is it underpins this linking of previous and current mental entities? The authors clearly see these 'entities' as being far from simple. There are contextual of relationships at play, ones which are familiar and arising from experience. It is the functionality of the cup in a somewhat elaborated mental picture which supports the recognition. Something which is not simple is brought into focus by detailed relationships over which we exercise a degree of mental ownership. How can start to model such relationships and what they deliver. How can we carry out the sort of structural comparisons between diverse contexts that support such analogy forming. We suspect a corresponding hierarchical construct, involving increasingly complex descriptions—and based in relationships that we have a basic grip of. What we do have a secure grip of is the level at which *categorization*, supported by *computational* criteria, comes into play. And the analogy which comes into play for us, is that of the emergent higher type (mathematically type-2) patterns in nature, which Turing deconstructed in terms of computable reaction-diffusion elements, and hence found descriptions of via appropriate differential equations. Of course, networks and their pattern-forming are extensively investigated nowadays according to all their fascinating specificities, and high-impact journals trace their surprising intricacies—as one does as one encounters real emergence. But surely there is more to do than experience the botanical garden, and make interesting observations. What are the overarching informational structures which host these wonders?

Faced with the incomputability of important relations, Turing introduced the notion of an oracleTuring machine, in order to compare degrees of incomputability. At a stroke, this gave us a structuring of information according to its relative computational complexity. The oracle machine computes just like the standard machine described earlier, but has an extra facility for occasionally asking a question about a finite part of a given real number provided as the oracle—the oracle being a repository of auxiliary information to be used during any given computation.

This idea gave us a model of how we compute using data given to us from unknown sources. It not only gave us a way of comparing computability of reals. It could also be used to model real world causality, where the mathematical law for computing applications of the law from one state to another was computable. This was a framework within which Newtonian computable laws fitted nicely, at the basic level. Looked at globally, one obtained a model of computable content of structures, based on partial computable (p.c.) functionals over the real numbers. So any physical context, characterised by computable relationships between pairs of reals representing particular physical states, could be presented as an instantiation of part of the resulting mathematical structure—often termed the Turing universe.

Strangely, despite Turing's later interest in interactive computation, he never returned to the oracle machine.

In 1948, Emil Post mathematically prepared the Turing universe for further investigation by grouping reals together into equivalence classes (called degrees of unsolvability) of inter-computable reals, the degrees ordered by the induced ordering got from relative computation. The resulting structure, over the years, was discovered to have a very rich infrastructure. Mathematically, what was seen as a high degree of pathology provided the means for defining a rich array of relations and individuals. The complexity, of course, was just what one might expect from a mathematical structure capable of reflecting causal structure from different physical contexts. The definable relations formed a mathematical counterpart of the richness of emergent phenomena in the real world.

By the 1960s, the theory of this basic structure—now termed the Turing degrees—was becoming a very active area of pure research, one of formidable technical complexity. And this leads us to what has become known as *Hartley Rogers' Programme*, originating with a paper of his on "Some problems of definability in recursive function theory", based on an invited talk at the 1965 Logic Colloquium in Leicester. The fundamental problem implicit in the paper and talk, was that of characterizing the Turing definable/invariant relations over the structure of the Turing degrees. The intuition was that these definable relations are key to pinning down how higher order relations on the real world can appear to be computed. An important observation being that the breakdown of definability potentially underpins ontological and epistemological ambiguities in the real world—such as confused mental states, or uncertainty at the quantum level in physics.

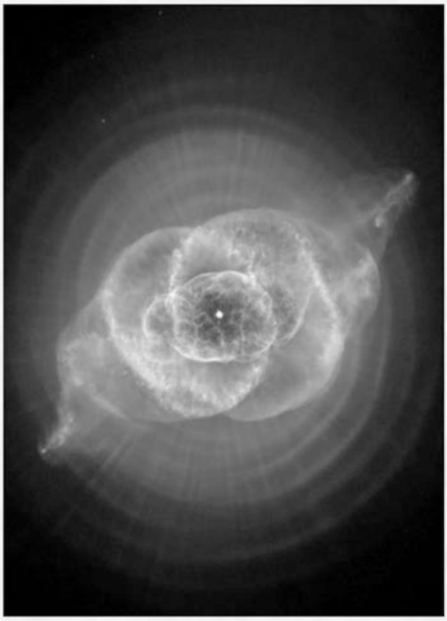


Fig. 10.9 Cat's eye nebula (NASA image, hubble telescope, see https://en.wikipedia.org/wiki/Cat%27s_Eye_Nebula

This aspect of physical computation—the Turing universe and emergent relations—is treated in more detail elsewhere (for example, in my (2013)). The key message is that in the real world one may describe global relations in terms of local structure, so capturing the computational basis of large-scale phenomena. And that mathematically this can be formalized as definability, or more generally invariance under automorphisms, over structure based on relative computation.

This is just one important ingredient in liberating us from the computational flatness of real world information. Its role is particularly appropriate to the scientific context, with clearly observed counterparts to the computability of basic relations, representable at accessible type levels. Here, the correspondence between the levels of abstraction, over which Hofstadter and Sander's analogies play out, fits well with the mathematical type structure. Scientifically, one may need to focus on local instantiations of context, but the relationship to the mathematics is both useful and informative (Fig. 10.9).

Over the sort of information the human brain deals with, one has difficulty isolating closed systems, and the selection of pertinent relations is essentially heuristic and shifting in particulars. Analogies weaken to metaphors, and the grasp of computers becomes at best statistical, made fallible by the patchy reducibility of informational type. What we are left with is a better understanding of the underlying patterns of knowing and not knowing, and appreciation of the appropriate relationship between the algorithmic and more characteristically *human* thought. Most importantly, we properly situate digital computation within the wider computational context, and rediscover the role of language in navigating our increasingly real sense of the typing of embodied information.

Acknowledgments This article is based on an invited talk at *Turing 100: A Celebration of the Life and Legacy of Alan Turing*, at Boston University, Monday, November 12, 2012. Supported by the John Templeton Foundation research grant: *Mind, Mechanism and Mathematics*, July 2012–June 2015.

References

Aaronson, S. 2013. *Quantum Computing Since Democritus*. Cambridge: Cambridge University Press.

Abramsky, S. 2013. Two Puzzles About Computation, in Cooper and van Leeuwen (eds.) (2013), pp. 53–57.

Adamatzky, A., et al. 2013. On Creativity of Slime Mould. *International Journal of General Systems* 42: 441–457.

Arkin, R.C. 1998. *Behavior-Based Robotics*. Cambridge, MA: MIT Press.

Beckermann, A., H. Flohr, and J. Kim. (eds.), 1992. *Emergence or Reduction? – Essays on the Prospects of Nonreductive Physicalism*. Berlin: de Gruyter.

Blackmore, S. 1999. *The Meme Machine*. New York: Oxford University Press.

Blum, L., F. Cucker, M. Shub, and S. Smale. 1997. *Complexity and Real Computation*. New York: Springer.

Broad, C.D. 1923. *The Mind and Its Place in Nature*. London: Kegan-Paul.

Brooks, R. 2001. The Relationship Between Matter and Life. *Nature*, Insight Feature 409, 18 January 2001, 409–411.

Claude, C.S., and K. Svozil. 2008. Quantum Randomness and Value Indefiniteness. *Advanced Science Letters* 1 (2): 165–168.

Cooper, S.B. 2004. *Computability Theory*. New York: Chapman & Hall/CRC.

———. 2011. From Descartes to Turing: The Computational Content of Supervenience. In *Information and Computation*, ed. M. Burgin and G. Dodig-Cmkovic. Singpore: World Scientific Publishing.

———. 2013. The Mathematician's Bias – And the Return to Embodied Computation, in Zenil, H. (ed.) (2012), pp. 125–142.

Cooper, S.B., and J. van Leeuwen (eds.). 2013. *Alan Turing: His Work and Impact*. New York: Elsevier. Reprints of many essays of Turing's with commentary.

Damasio, A. 1999. *The Feeling of What Happens: Body and Emotion in the Making of Consciousness*. London: Harcourt Brace.

Davis, M. 2000/2011. *The Universal Computer: The Road from Leibniz to Turing*. New York: W.W. Norton/AK Peters/CRC/Taylor & Francis.

Dennett, D. 1991. *Consciousness Explained*. New York: Little Brown.

———. 2013. Turing's "Strange Inversion of Reasoning", in Cooper and van Leeuwen (eds.) (2013), 569–573.

Denning, Peter J., and P. Wegner. 2010. *Ubiquity* Symposium, 'What is Computation?', Editor's Introduction, *Ubiquity, an Association for Computing Machinery Publication*, October, 1. doi:10.1145/1865907.1870596.

Deutsch, D. 1985. Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. *Proceedings of the Royal Society of London, Series A* 400: 97–117.

———. 1997. *The Fabric of Reality*. New York: Penguin Books.

———. 2006. "Questions and Answers with David Deutsch", Readers' Q & A. In *New Scientist Science in Society*, 21 December 2006, at http://www.newscientist.com/article/dn10691-readers-q--a-with-david-deutsch.html?full=true#.VNfollPF_HQ.

Downey, R.G., and D.R. Hirschfeldt. 2010. *Algorithmic Randomness and Complexity*. New York: Springer.

Earman, J. 1986. *A Primer on Determinism*. Western Ontario Series in Philosophy of Science, New York: Springer.

Einstein, A. 1950. *Out of My Later Years*. New York: Philosophical Library.

———. 1969. Autobiographical Notes, in P.A. Schlipp (ed. and trans.), *Albert Einstein: Philosopher-Scientist*, 3rd. ed. (1st. ed. 1949), Open Court Publishing, 1–95.

Feferman, J. 1988. Turing in the Land of O(z), in *Herken* (ed.) (1988), 113–148.

Floridi, L. 2011. *The Philosophy of Information*. Oxford: Oxford University Press.

Floyd, J. 2013. Turing, Wittgenstein and Types: Philosophical Aspects of Turing's 'The Reform of Mathematical Notation and Phraseology' (1944–5), in Cooper and van Leeuven (eds.) (2013), 250–253.

Fortnow, L. 2010. What is Computation? The Enduring Legacy of the Turing Machine. *Ubiquity, an Association for Computing Machinery Publication*, December, 5, doi:10.1145/1895419.1921573.

Frailey, D.J. 2010. What is Computation? Computation is Process. *Ubiquity, an Association for Computing Machinery Publication, November*. doi:10.1145/1880066.1891341.

Gödel, K. 1944. Russell's Mathematical Logic, in Schilpp, P.A. (ed.) (1951), *The Philosophy of Bertrand Russell* (3rd edition), 123–153). New York: Tudor.

Guth, A.H. 1997. *The Inflationary Universe: The Quest for a New Theory of Cosmic Origins*. New York: Perseus.

Halmos, P.R. 1987. *I Have a Photographic Memory*. Providence: American Mathematical Society.

Hadamard, J. 1945. *The Psychology of Invention in the Mathmatical Field*. Princeton: Princeton University Press.

Haigh, T. 2013. 'Stored Program Concept' Considered Harmful: History and Historiography, in Paola Bonizzoni, P., Vasco Brattka, V., and Benedikt Löwe, B. (eds.), *The Nature of Computation. Logic, Algorithms*, 241–251, Proceedings of the 9th Conference on Computability

in Europe, CiE Milan, Italy, July 2013, Lecture Notes in Computer Science 7921, Berlin/Heidelberg: Springer.

Herken, R., ed. 1988. *The Universal Turing Machine: A Half-Century survey*. New York: Oxford University Press.

Hilbert, D., and W. Ackermann. 1928. *Grundzüge der Theoretischen Logik*. Berlin: Springer.

Hodges, A. 1983/2012. *Alan Turing: The Enigma of Intelligence*. London: Vintage.

Hofstader, D., and E. Sander. 2013. *Surfaces and Essences: Analogy as the Fuel and Fire of Thinking*. New York: Perseus Books.

Kim, J. 1998. *Mind in a Physical World*. Cambridge, MA: MIT Press.

———. 2005. *Physicalism, or Something Near Enough*. Princeton: Princeton University Press.

Kreisel, G. 1970. Church's Thesis: A Kind of Reducibility Axiom for Constructive Mathematics. In *Intuition and Proof Theory: Proceedings of the Summer Conference at Buffalo, N.Y. 1968*, ed. J. Kindo, J. Myhill, and R.E. Vesley, 121–150. Amsterdam: North-Holland.

Kuhn, T.S. 1996. *The Structure of Scientific Revolutions*. 3rd ed. Chicago: University of Chicago.

Leibniz, G.W. 1714. The Principles of Nature and Grace, Based on Reason, in Ariew, R. and Garber, D., (eds. and trans.) *G.W. Leibniz, Philosophical Essays,* 206–213. Indianapolis: Hackett Publishing.

Leijonhufvud, A. 2010. The Nonlinear Path of Macroeconomic Dynamics. In *Computable, Construtive, and Behavioural Economic Dynamics: Essays in Honour of Kumaraswamy Velupilai*, ed. S. Zamelli, 62–69. New York: Routledge.

Longley, J. 2005. Notions of Computablity at Higher Types I. In *Logic Colloquium 2000*, *Lecture Notes in Logic 19 ASL*, eds. R. Cari, A. Razborov, S. Todorcevic, and C. Wood, 32–142. A.K. Peters: Wesley.

Maini, P.K. 2013. Turing's Theory of Morphogenesis, in Cooper and Leeuven (eds.) (2013), pp. 684–688.

Mardia, K.V., and S.B. Cooper. 2012. Alan Turing and Enigmatic Statistics. *Bulletin of the Brasilian Section of the International Society for Bayesian Analysis* 5 (2): 2–7.

McCulloch, W., and W. Pitts. 1943. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics* 5: 115–133.

McGilchrist, I. 2009. *The Master and His Emissary: The Divided Brain and the Making of the Western World*. New Haven: Yale University Press.

McLaughlin, B.P. 1992. The Rise and Fall of British Emergentism, in Beckermann, Flohr and Kim (eds.), pp. 49–93.

McLaughlin, B., and Bennett, K. 2014. Supervenience. *Stanford Encyclopedia of Philosophy* http://plato.stanford.edu/archives/spr2014/entries/supervenience/.

Murray, J.D. 2013. After Turing – The Birth and Growth of Interdisciplinary Mathematics and Biology, in Cooper and Leeuven (eds.) (2013), 739–752.

Penrose, R. 1987. Quantum Physics and Conscious Thought. In *Quantum Implications: Essays in Honour of David Bohm*, ed. B.J. Hiley and F.D. Peat, 105–120. New York: Routledge.

Pinker, S. 1997. *How the Mind Works*. New York: W.W. Norton.

Poincaré, H. 1921. *The Foundations of Science: Science and Hypothesis, The value of Science, Science and Method.* New York: The Science Press. Reprinted 1982, Washington, DC: University Press of America.

Post, E. 1948. Degrees of Recursive Unsolvability: Preliminary Report (abstract). *Bulletin of the American Mathematical Society* 54: 641–642.

Putnam, H. 1960. Minds and Machines. In *Dimensions of Mind*, ed. S. Hook. New York: New York University Press. Reprinted in Putnam 1975, 362–385.

———. 1967/1975. The Nature of Mental States, in Putnam (1975), 429–440. Originally published as "Psychological Predicates" in Capitan, W.H., and Merrill, D.D. (eds.), *Art, Mind and Religion*, 37–48. Pittsburgh: University of Pittsburgh Press (1967).

———. 1975. *Mind, Language, and Reality*. Cambridge: Cambridge University Press.

———. 1984. Is the Causal Structure of the Physical Itself Something Physical? In *Causation and Causal Theories*. *Midwest Studies in Philosophy*, vol. 9, ed. P.A. French, T. Uehling, and

H. Wettstein, 3–16. Minneapolis: University of Minnesota Press. Reprinted in Putnam (1990), 80–95.
———. 1988a. *Representation and Reality*. Cambridge, MA: MIT Press.
———. 1988b. Why Functionalism Didn't Work, in Putnam (1988a), 73–89.
———. 1990. *Realism with a Human Face*. Cambridge: Harvard University Press.
Reichenbach, H. 1924. *Axiomatik der relativistischen Raum-Zeit-Lehre.* Trans. M. Reichenbach as *Axiomatization of the Theory of Relativity,* University of California Press, 1969.
Rogers, H. 1967. Some Problems of Definability in Recursive Function Theory. In *Sets, Models, and Recursion Theory* (Proc. Summer School Math. Logic and Tenth Logic Colloquium, Leicester, 1965), 183–201. Amsterdam: North-Holland.
Smolensky, P. 1988. On the Proper Treatment of Connectionism. *Behavioral and Brain Sciences* 11: 1–23.
Smolin, L. 2006. *The Trouble with Physics: The Rise of String Theory*. New York: Houghton Mifflin.
Soare, R.I. 1987. *Recursively Enumerable Sets and Degrees*. New York: Springer.
Taleb, N. 2005. *Fooled by Randomness*. 2nd ed. New York: Random House & Penguin.
———. 2007. *The Black Swan: The Impact of the Highly Improbable.* London: Allen Lane (2nd ed., 2010).
Tarski, A. 1931. *Sur les ensembles définissables de nombres reels I. Fundamenta Mathematicae,* 17, 210–239. English translation "On Definable Sets of Real Numbers" in Tarski (1983), 110–142.
———. 1983. *Logic, Semantics, Metamathematics.* J. Corcoran (ed.), J.H. Woodger (trans.), 2nd revised ed., Indianapolis: Hackett.
Tegmark, M. 2003. Parallel Universes. *Scientific American* 288 (5): 40–51.
Turing, A.M. 1936/37. On Computable Numbers with an Application to the *Entscheidungsproblem. Proceedings of the London Mathematical Society* 2, 42, 230–265. Correction, *Proceedings of the London Mathematical Society* 43, 544–546. Reprinted with correction in Cooper and van Leeuwen (eds.) (2013), pp. 16–43.
———. 1939. Systems of Logic Based on Ordinals. *Proceedings of the London Mathematical Soci*ety 2, 45:161–228. Reprinted in Cooper and van Leeuwen (eds.) (2013), 151–197.
———. 1947. Lecture to the London Mathematical Society 20 February 1947. In *Turing 1992*, 87–106. Originally published in B.E. Carpenter, and R.N. Doran (eds.), A.M. Turing's ACE Report and Other Related Papers (1986, MIT Press: 106–124); reprinted in Cooper and van Leeuven (eds.) (2013), 486–498.
———. 1952. The Chemical Basis of Morphogenesis. *Philosophical Transactions of the Royal Society of London* B 237: 37–72. Reprinted in Cooper and van Leeuwen (eds.) (2013), 689–722.
———. 1954. Solvable and Unsolvable Problems. *Science News* 31: 7–23. Reprinted in Cooper and van Leeuwen (eds.) (2013), 322–331.
———. 1992. *Collected Works of A.M. Turing: Mechanical Intelligence*. D.C. Ince (ed.), Amsterdam: North Holland.
Velupillai, K.V. 2005. The Unreasonable Ineffectiveness of Mathematics in Economics. *Cambridge Journal of Economics* 29: 849–872.
Velupillai, K.V., S. Zambelli, and S. Kinsella, (eds.). 2011. *Computable Economics*. Northampton: Edward Elgar Publishing.
Webb, J.C. 1980. *Mechanism, Mentalism, and Metamathematics – An Essay on Finitism*, Synthese Library. Dordrecht/Boston: Reidel Publishing Co..
Weinberg, S. 2002. Is the Universe a Computer?. *The New York Review of Books,* 24 October 2002.
Wittgenstein, L. 1921. *Tractatus Logico-Philosophicus.* Trans. C.K. Ogden. London: Routledge & Kegan Paul, reprint with corrections, 1931, republished 1981. First German edition in *Annalen der Naturphilosophie* 14, W. Ostwald (ed.), Leipzig, 1921, 12–262.
Woit, P. 2006. *Not Even Wrong—The Failure of String Theory and the Continuing Challenge to Unify the Laws of Physics*. London: Jonathan Cape.
Zeil, H. 2013. *A Computable Universe: Understanding and Exploring Nature as Computation*. Hackensack/London: World Scientific Publishing.

Part III
Human, Machine, and Mind

Chapter 11
On Computing Machinery and Intelligence

Patrick Henry Winston

Abstract The Turing Test has been part of the lexicon of artificial intelligence ever since Turing proposed it in his famous paper, "Computing Machinery and Intelligence" (1950). Close reading suggests, however, that Turing's real purpose in writing the paper was, firstly, to attack the arguments of skeptics so as to establish that there is no reason to believe computers cannot be intelligent, and secondly, to propose a program of research. Today, Turing would likely have structured his thinking differently, and perhaps focused on different questions, perhaps the questions on which I focus: What is it that makes us different from all other species? And what is it that we have in common with other species that makes the difference important? I conclude that story understanding makes us different and that story understanding rests on directed perception. I elaborate on story understanding, explaining how a simple substrate of English analysis, common sense inference, and concept search enable the Genesis story understanding system to demonstrate a range of competences, including culturally grounded story interpretation and question-driven analysis. All this leads to a discussion of open questions and a reassessment of Turing's paper's fundamental contribution.

Alan Turing's paper, *Computing Machinery and Intelligence* (1950), introduced what he called the imitation game and what we have all come to call the Turing test. Everyone seems to know about the test, but curiously, few seem to have read the paper, even though many consider it, without reading it, to anchor modern efforts to develop a computational account of human intelligence.

I argue that Turing's fundamental contribution was not the test, but rather the message, *computers can be intelligent*. I buttress my argument by noting that although Turing's message launched decades of value-producing research in artificial intelligence, we still lack a computational account of human intelligence. I suggest that progress has been slow because of insufficent attention to what makes

P.H. Winston (✉)
Massachusetts Institute of Technology, Room 251, 32 Vassar Street, Cambridge, MA 02139, USA
e-mail: phw@mit.edu

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_11

human intelligence special, and then I argue that what makes human intelligence special is our unique ability to understand, tell, and manipulate stories. I explain how thinking can be viewed as story understanding using the plot from Shakespeare's *Macbeth* to illustrate the basic ideas. I then show how the basic ideas enable culturally grounded interpretation and question-driven analysis in the context of a murder story read differently by Asian and American readers. Such capabilities lead me to claim that reasoning, the focus of Turing's research proposal, is best viewed as a special case of story understanding. I conclude with open questions and a salute to Turing's vision.

11.1 The Objections

When you actually read Turing's paper, you see that very little of it is focused on the test, just three pages in all. The rest is devoted to explaining how computers work (six pages), a suggested program of research (five pages), a prediction (just a few words), and importantly, ten pages offering nine counterarguments to arguments against the possibility of computers becoming intelligent.

The test, Turing wrote, stands in place of a definition of intelligence, which he considered a hopeless enterprise. He proposes instead the idea of a human interrogator questioning a man and a computer, at a distance, in typed English, and if the probability of the interrogator correctly identifying the computer, after 5 min, is less than 70%, the computer has passed the test.

I have my students read Turing's paper in a subject I teach, so I read it myself once a year. Each time, I become more convinced that Turing was not really focused on the challenge, but rather with getting the question, *what is intelligence*, out of the way, so that he could focus on his real interest, which was whether there are limits to the degree to which a computer can be intelligent, whatever intelligence is. To that end he examined the popular objections of the day:

1. The theological objection.
2. The "Head in the sands" objection.
3. The mathematical objection.
4. The argument from consciousness.
5. The argument from various disabilities.
6. Lady Lovelace's objection.
7. The argument from continuity in the nervous system.
8. The argument from informality of behavior.
9. The argument from extrasensory perception.

Objection five, the argument from various disabilities, is among those that persist. Many align themselves with the idea that no matter how impressive IBM's Watson has become at playing Jeopardy!, or no matter how accurately Apple's Siri responds to our spoken requests, there will remain thoughts and actions forever beyond computers. Turing cited these as representative of those commonly offered:

> Be kind, resourceful, beautiful, friendly, have initiative, have a sense of humor, tell right from wrong, make mistakes, fall in love, enjoy strawberries and cream, make some one fall in love with it, learn from experience, use words properly, be the subject of its own thought, have as much diversity of behavior as a man, do something really new.

About 20 years after Turing published "Computing Machinery and Intelligence" (1950), Seymour Papert coined the term *the unthinkability fallacy*, a label for failures of imagination that lead to claims about what computers cannot ever do. In a Philippic against artificial intelligence, *Alchemy and Artificial Intelligence*, Hubert Dreyfus boldly wrote about the weakness of the chess-playing programs of the day (1965). In his rebuttal, *The Artificial Intelligence of Hubert Dreyfus: a Budget of Fallacies* (1968), Papert delighted in reporting that in a challenge match against Richard Greenblatt's chess-playing program, Dreyfus was "roundly trounced," all this under the heading, *Computers Can't Play Chess*, and the subheading, "Nor can Dreyfus." Eventually, in 1997, IBM's Deep Blue program beat Garry Kasparov, the human champion, settling the matter unequivocally, if not uncontroversially. Deep Blue was disassembled even as Kasparov called for a rematch, and besides, the methods employed by Deep Blue in no way model human chess thinking, but rather rely on brute force search.

An example of the unthinkability fallacy was at work in objection six, Lady Lovelace's Objection. As quoted by Turing, Lady Lovelace, said to be Charles Babbage's programmer, was thinking of whether a computer could do anything on its own:

> The Analytical Engine has no pretensions to originate anything. It can do whatever we know how to order it to perform.

We should forgive Lady Lovelace for this, of course, because at the time it was unthinkable that a computer might someday learn from spoken instruction, or from that which is presented to its sensors, or better still, from its own experiences as it wanders through our physical world on robotic legs or as it extracts knowledge from cyberspace. Now, all are well within the state of the art.

Possibly the strangest objection was objection nine, the argument from extrasensory perception. Evidently, Turing believed in what has become rarely believed:

> I assume that the reader is familiar with the idea of extra-sensory perception, and the meaning of the four items of it, viz. telepathy, clairvoyance, precognition and psycho-kinesis. These disturbing phenomena seem to deny all our usual scientific ideas. How we should like to discredit them!

How is it possible that a mathematical genius would believe in extra-sensory perception? My answer is that he believed because he wanted to, possibly because he wanted to believe in the occult in general, possibly because he wanted to communicate with a deceased friend (see chapter 2 of Hodges's biography (2012)).

That sort of want-to-believe is also evident when skeptics suppose that humans are somehow exempt from constraints on computation. Objection three, the mathematical objection, runs along these lines: Turing himself proved that there is no algorithm that can decide, for all possible program-input pairs, whether or not the program will run forever without stopping. The objectors happily acknowledge that

the proof limits what can be done by an algorithm, hence by a computer, but strangely suppose people are not subject to the same proof, thus arguing by assumption that people are superior to computers.

Of course, skeptics are not easily swayed by attacks on their positions. After all, Turing might have left out, overlooked, or failed to anticipate something crucial. Or, a skeptic can find a way to dismiss one of Turing's arguments, perhaps adhering firmly to one of the disabilities, taking that as sufficient evidence to bring down Turing's entire position. Turing acknowledged all that:

> The reader will have anticipated that I have no very convincing arguments of a positive nature to support my views. If I had I should not have taken such pains to point out the fallacies in contrary views.

Although Turing believed he had no convincing arguments in favor of computer intelligence, he did speculate on what should be done to make computers intelligent:

> Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain.

But how exactly would this be done? Turing's thinking was influenced by what he was, what he could imagine, and what he had proposed. Because Turing was a mathematician, it is not surprising that he focused on reasoning, not on perception. Accordingly Turing saw little need for his child computer to have a body: "We need not be too concerned about the legs, eyes, etc. The example of Miss Helen Keller shows that education can take place provided that communication in both directions between teacher and pupil can take place by some means or other."

Moreover, the possibility of a body was only barely imaginable in Turing's day: "It will not, for instance, be provided with legs, so that it could not be asked to go out and fill the coal scuttle. Possibly it might not have eyes."

And what Turing had proposed was a test during which communication with the human player and with the computer player would be via typed text. There would be no possibility of the interrogator holding up an object and saying "what's this?" Again, there would be no need for perception.

11.2 The Way the World Became

Turing's paper established that serious people can think seriously about intelligent computers. In 1957, just a few years later, a lot of serious people came together at the Dartmouth conference determined, according to the funding proposal, "… to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves." Then, Marvin Minsky published *Steps toward Artificial Intelligence* (1961), which elaborately surveyed the work that had been done so far and proposed what ought to be done next.

For a little more than a decade, progress was so rapid, it seemed that computers soon would exhibit human level intelligence, and even super-human intelligence. And possibly, it might be that a computer could be set on the problem of making itself smarter, thus leading to a kind of chain reaction, already mentioned in Turing's paper and popularized today by Ray Kurzweil as *the singularity*, the subject of a forward to the third edition of John von Neumann's classic book, *The Computer and the Brain* (2012).

Great laboratories were formed at MIT by Minsky, at Stanford by John McCarthy, and at Carnegie-Mellon by Alan Newell and Herbert Simon. In Minsky's laboratory at MIT, students wrote programs that performed symbolic integration, solved geometric analogy problems, analyzed line drawings of block structures, engaged in simple dialogs about block structures, and even learned something definite from each example of an arch and an arch near miss presented by a knowledgeable teacher.

Alas, while impressive, the programs were narrowly focused and fragile, illustrations more than demonstrations. In frustration, the field eventually splintered into subfields, which include neural nets, genetic algorithms, statistical inference, machine learning, vision, and language, each with its own dogma.

The field also became more applications oriented, so much so that today it is hard to find a big computer system not enabled in part by research conducted in artificial-intelligence laboratories. Many of these systems do their work inconspicuously, such as GateKeeper, a system that determines the gate at which airplanes are parked; others are very conspicuous, such as Siri, which will happily tell you how to get to the airport. All these are not so fragile as the programs of the 1960s and 1970s, but they are still narrowly focused.

But are they intelligent? What is intelligence? Are we driven back to Turing's test? Do today's systems pass narrowed forms of Turing's test? What would Turing think?

Perhaps Turing today would dismiss these questions about intelligence in virtue of the light Minsky has shed on the nature of the word in *The Emotion Machine* (2006). Minsky points out that *intelligence*, along with words such as *emotion*, *creativity*, and *intuition*, are what he calls suitcase words, words that cover such a broad range of concepts you can fit whatever you want into them. From this perspective, if you ask if GateKeeper or Siri are intelligent, the easy answer is sure, they are intelligent, even superintelligent, in some ways, but not in all the ways exhibited by humans.

What then are those special ways so far exhibited only by humans? A look backward provides interesting perspective.

11.3 The Way the World Was

In the nineteenth century, Lady Lovelace thought about whether computers could think. Two and a half millennia before, Plato thought about thinking.

More importantly, from the perspective of understanding intelligence, we as a species started thinking symbolically not millions of years ago but only about 60,000 years ago. It seems to have happened all of the sudden, long after we had achieved our modern anatomical form, and some, notably the paleoanthropologist Ian Tattersall (1998), believe what happened should be viewed as a kind of discovery of how we could use what was already in place in our brains, rather than a slow, evolving capability.

Before the big change, whatever it was, something was missing. We had trouble innovating. We could make stone tools, but our technology for making stone tools did not change for many thousands of years. After the big change, we soon, in evolutionary time, we turned seashells into necklaces. We carved figurines. We started painting caves, and we painted them in plainly symbolic ways.

Noam Chomsky, equipped to put more substance into what symbolic thinking means, suggests that we developed the ability to put two concepts together to make a third, and we can do this without disturbing the two component concepts, and we can do this without limit (2005).

Thus, we can describe, and among the things we can describe are events, and once we can describe events, we can string them together into stories, and once we have stories, we can use them to explain, to predict, to control, and we can blend them to form new stories, exercising a kind of creativity, and that, I believe, is what makes us different, and leads perhaps to an approach different from that of Turing.

11.4 The Way the World Will Be

Turing asked, what would it take to pass his test. I ask, what is it that makes us different from all other species, and what is it that we have in common with other species that makes the difference important. These questions lead me to two hypotheses:

The strong story hypothesis: The mechanisms that enable us humans to tell, understand, and recombine stories separate our intelligence from that of other primates.

The directed-perception hypothesis: The mechanisms that enable us humans to direct our perceptual faculties to answer questions, using real and imagined events, separates our intelligence from that of other primates.

Accordingly, my first aim is to develop a computational understanding of stories because I think that story understanding is the great differentiator. I also think stories are everywhere in human development and education as throughout life we are awash in fairy and folk tales, religious parables, ethnic narratives, history, literature, and news. And beyond that, we specialize in the stories of law, business, medicine, defense, diplomacy, and engineering, all of which are taught in large degree by case study.

And because I believe that you cannot fully understand unless you can build, my students and I have built the Genesis story understanding system (2011, 2012a, b). Lapsing momentarily into the vernacular, Genesis deploys common-sense rules and concept patterns, which enable Genesis to understand stories consisting of a few to a hundred or so simple sentences expressed in English. Representative examples include simple renderings of plots drawn from Shakespeare's plays, Grimm's fairy tales, Crow folklore, and accounts of cyber and kinetic warfare.

My second aim, not at all in Turing proposed plan, is to develop a computational understanding of perception, because I believe much of what we know is generated as we need to know it using our perceptual faculties on real and imagined events. Turing seems to have entirely missed the important contribution of perception in generating knowledge on the fly, as needed.

Say, "John kissed Mary." Ask, "Did John touch Mary?" Everyone knows the answer and everyone thinks they know it because they imagine an event and read the answer off of the imagined event with perceptual apparatus. Ask, "Does the sun ever set north of west?" The answer is readily forthcoming if an orange is available for experiment.

11.5 Thinking Viewed as Story Understanding

Turing seems to have thought thinking was a matter of reasoning, a symbolic process, a way of modeling thought that was very familiar to the ancient Greeks. I focus on story understanding, also a symbolic process. So how is Turing's focus different from mine? To answer that question, it is helpful to look a little more closely at what Genesis does.

First, Genesis uses syntactic and semantic analyzers to translate simple stories, expressed in a few dozen sentences, into an inner language. Being fond of Shakespeare, I often illustrate with a highly simplified version of the Macbeth plot; here is a portion, about 10% of the story told to Genesis:

> Lady Macbeth is evil and greedy. Duncan is the king, and Macbeth is Duncan's successor... Witches predicted that Macbeth will become king... Macbeth wants to become king because Lady Macbeth persuaded Macbeth to want to become the king. In order to murder Duncan, Macbeth murders the guards and Macbeth stabs Duncan. Macbeth becomes king... Macbeth's murdering Duncan leads to Macduff's fleeing to England... Macduff's fleeing to England leads to Macbeth's murdering Lady Macduff. Macbeth hallucinates at a dinner. Lady Macbeth says he hallucinates often... Lady Macbeth kills herself… Macduff kills Macbeth.

Genesis uses common-sense rules to make inference obvious to any human. Two such rules, expressed in English, indicate the consequences of murder: *if x murders y, then x harms y, and if x murders y, then y becomes dead*.

Using such rules, Genesis produces an elaboration graph, such as the one shown in Fig. 11.1 that captures how explicit events are connected to each other and to implicit events.

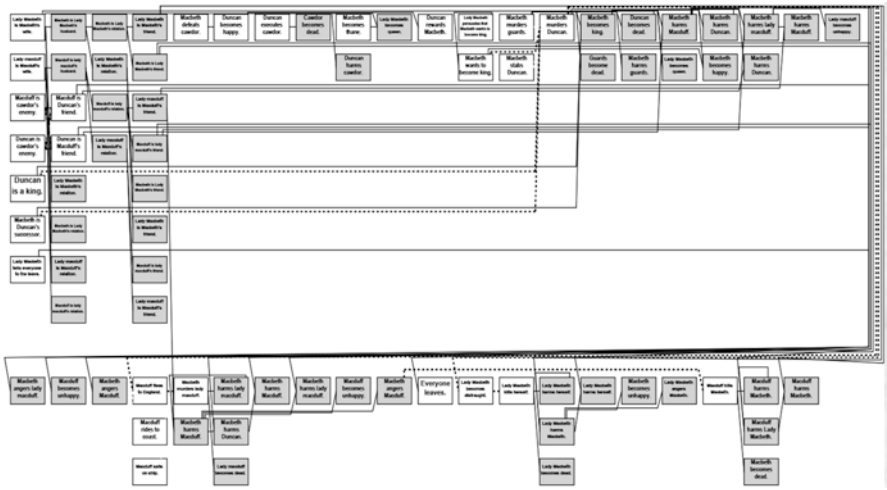


Fig. 11.1 Shown in birds-eye view, elaboration graphs give a sense of size and content. *White boxes* hold relations and events explicitly told in a simple summary of the Macbeth plot. *Gray boxes* hold conclusions generated by inference rules

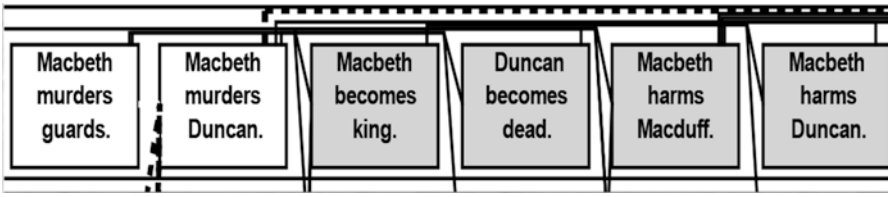


Fig. 11.2 Macbeth's murders have consequences. Duncan becomes dead. He is harmed

Zooming in on the consequences of Macbeth's murders, you see the connections in Fig. 11.2.

Other rules provide explanations. We humans like events to be explained, so when explanations are lacking, we often speculate: *If x angers y, y may kill x*. See, for example, the explanation shown in Fig. 11.3.

Once a story is read, Genesis uses concept patterns to examine its own conclusions, as captured in the elaboration graph. One such pattern captures a simple form of revenge: *Revenge occurs when x's harming y leads eventually to y's harming x*.

Genesis finds an instance of revenge in the elaboration graph shown in Fig. 11.1 because Macbeth's harming Macduff (by killing his wife) leads to Macduff harming Macbeth, as shown the elaboration graph fragment shown in Fig. 11.4.

Surprisingly, at least to me and the rest of Genesis's creators, commonsense rules and concept patterns enable Genesisto answer questions about what happened and why, reflect differing allegiances and cultural points of view, manifest consequences of personality traits, anticipate trouble, determine conceptual similarity, align stories for analogical reasoning, summarize, teach, and persuade (Winston, 2014).

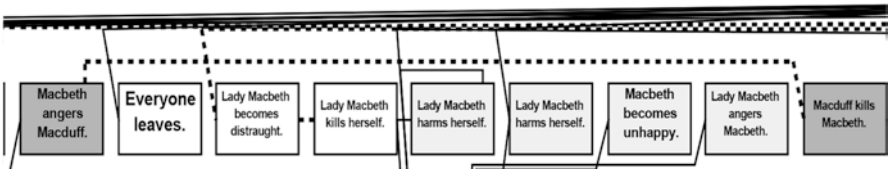


Fig. 11.3 We crave explanations, and if none are explicitly provided, we look for likely causes. The elaboration graph shows that Macduff may have killed Macbeth because Macbeth angered Macbeth. Fortunately, we do not always kill people who anger us, but it does happen, and in the absence of any other explanation, we are drawn to that one

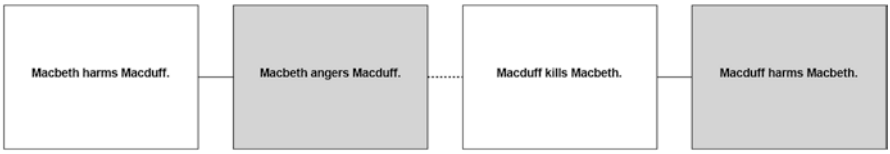


Fig. 11.4 Genesis reflects on its own inferences, looking for continues paths from one event to another. When one person harms another, and that leads to the second person harming the first, Genesis notes a simple form of *revenge*

One example of such sophistication was worked out by Hiba Awad (2013), whose work is based on cross cultural human experiments by Michael W. Morris and Kaiping Peng (1994). Morris and Peng found that Asian high-school students had a greater tendency to view violence as a consequence of the situations in which individuals are immersed, whereas American high-school students had a greater tendency to view violence as a consequence of the dispositions of the participating individuals. In a classic experiment, subjects are told a story about Lu, a Chinese student studying in America. Events happen, concluding with a shooting rampage.

Awad has Genesis read a version of the story using both a model of an Asian leaning reader and a model of a Western leaning reader. Both versions see *suicide*, because Lu kills himself in the end, and nothing else. But then both readers are asked "Did Lu kill Shan because America is individualistic?"

Now *America is individualistic* is not in the story as told, so the answer must be "No," or "I don't know." But in Awad's system, both readers inspect their models of themselves to see if they believe America is individualistic. The Asian reader's model does believe it; the Western reader's model does not. So, the Asian reader inserts *America is individualistic* into the story, and as shown in the Fig. 11.5, the new insertion triggers a chain of inferences starting with *America is corrupt*, moving on to *Lu becomes murderous*, and ending with *Lu kills Shan*. Genesis now sees several instances of the *Murderous influence* concept, one of which connects *America is individualistic* with *Lu kills Shan*, enabling the Asian reader to answer affirmatively; the Western reader's model does not find *America is individualistic* in memory, and so answers negatively. Thus, Awad shows how Genesis can model cultural bias and how a question can lead a Genesis reader to a conclusion not otherwise reached.

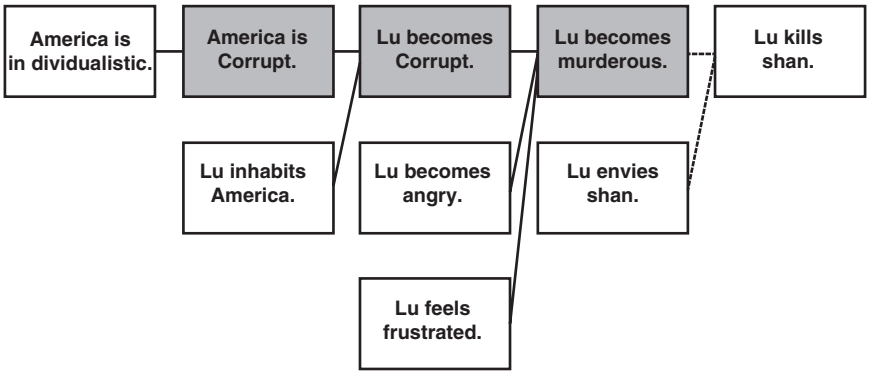


Fig. 11.5 Human asks, Did Lu kill Shan because America is individualistic? Genesis replies, "It looks like the Asian reader of *Lu murder story/eastern*, **on reflection, believes** America is individualistic, which **enables him to believe** Lu kills Shan because America is individualistic."

In doing its work, Genesis exhibits aspects of all the reasoning levels cited by Minsky in *The Emotion Machine* (2006), because Genesis has instinctive reactions (built in if-then rules), learned reactions (learned if-then rules), deliberative thinking (inspection of its own reasoning driven by desired to establish cause), reflective thinking (deployment of concept patterns), self-reflecting thinking (model of rules and concepts of self), and self-conscious reflection (models of rules and concepts of others).

Genesis also can be made to spot acts of kindness, be resourceful, tell right from wrong, make mistakes, learn from experience, use words properly, and be the subject of its own thought. Genesis does all this at an illustration level, not at a robust human level, but interestingly, all these are among the supposed impossibilities Turing attributed to the skeptics in objection 5, the argument from various disabilities.

So how is Turing's focus on reasoning different from my focus on story understanding? Turing did not mention story understanding and probably would have reached a quick conclusion that if you can reason, you can deploy your reasoning faculty to understand stories. I think, however, he would have had it upside down. With all its capability, Genesis certainly can read and understand recipes, as recipes are sequences of prescribed events, which makes understanding a recipe a special case of story understanding. And one class of recipes dictate how to take axioms forward to conclusions and how to prove hypotheses by working backward toward known axioms. So what Turing focused on might just be shadows dancing on a wall, a special case of the story understanding capability that we acquired when we became symbolic.

11.6 Getting There

Our work on Genesis is superficially attacked by those who correctly observe that Genesis does not understand the nuances of Shakespeare, the brothers Grimm, and Native Americans. To be sure, these are only handy sources of familiar and exotic plots, which Genesis handles at a very elementary level. The point is not to demonstrate that Genesis performs at the level of an adult human, but rather to model, at an illustration level, what might lie at the core of human story understanding competence. It unreasonable to think that Genesis models nothing just because its understanding is not such that it can write papers for learned journals. Papert had a name for this kind of unreasonableness: he called it the superhuman fallacy, and pointed out that computers do not have to write plays as good as Shakespeare's to be considered intelligent, because if so, Shakespeare would be the only intelligent person ever, and even he would not be intelligent, because he never wrote symphonies as good as Mozart's.

A more serious critique is that knowing that killing implies that the killed becomes dead is to know hardly anything about what it means to kill. Surely what a human knows is effectively limitless.

But what a human knows is not effectively limitless. An often repeated calculation, consuming the back of many envelopes, speculates on an upper bound to the number of facts an adult might know. You simply suppose a person can acquire some number of facts in a day, perhaps one every 3 or 4 s, a 1000 an hour, for 10 h of concentrated effort every day, over, say, 20 years:

$$1000 \times 10 \times 365 \times 20 = 73,000,000$$

So, what can you know about killing? Divide the number by a few thousand, as a few thousand words in English cover a lot of ground. You get about 20,000. Now I think that is probably an overestimate by a factor of ten at least, but 2000 is still 200 times what Genesis knows about killing. How can such knowledge be provided to some future Genesis?

Perhaps this takes us back to Turing's child machine proposal, but today we can go back with technology Turing could not have conceived: crowd sourcing on the world-wide web. Volunteers, motivated and monitored for quality of effort, can produce a great deal of common sense knowledge and have done so in early experiments (Havasi et al. 2009).

Alas, even a million volunteers might not be up to the task, because there is a lot they might not think to say. Fortunately, however, what we have the capacity to know is far greater that what we have been told because of our on-the-fly capability to generate new knowledge, as needed, via inference, via operations on real and imagined perceptions, and via manipulation of real and imagined stories. From the story perspective, what we know about any action is determined by the ensemble of stories we have witnessed and have been told that involve that action. It becomes not a matter of how many facts can be crammed in during 20 years, but rather of how

many stories can have their essence usefully assimilated in 20 years. A few thousand perhaps? Genesis has not been optimized for speed, so it is not particularly fast, but it reads and processes about 50 words per second; in principle, it could read a typical novel or history book in half an hour. And of course putting 1000 computers to work on a problem has become common place, so a 1000-computer collection could, in principle, consume 1.75 million books per year.

What, however, is the *in principle* qualification? Alas, it signals that Genesis remains early basic research, which leads me to a discussion of open questions.

11.7 Open Questions

Artificial intelligence, from an applied perspective, has become a large, diverse field, with many subfields, each with many open questions. I limit myself here to science-side open questions focused on understanding human intelligence.

I expect that the hardest questions are those we have not yet come across, but for now, at least, the strong story hypothesis and the directed perception hypothesis both provide questions.

Story understanding offers many of small and medium size. Representative examples from our current list of two dozen are: how can Genesis learn from contrasting outcomes, and how can Genesis discover common sense rules and concept patterns automatically?

The great breakthrough-requiring question is: how can Genesis graduate from the language of simple, straightforwardly expressed story summaries to the language adults use in communicating with other adults. It is sobering to compare what Genesis can understand with real literature. I sobered myself one day by looking at Dickens's *David Copperfield*. When we write for Genesis, our simple sentences rarely contain much more than ten words; in Dickens's *David Copperfield* complex sentences, with many subordinate clauses, of 80 or 100 words are common. When we write for Genesis, we use no dialog; *David Copperfield* is full of it. When we write for Genesis, we help understanding along with frequent sentences containing *because* or *leads to*; *David Copperfield* offers little such help, and cause-suggesting constructions rarely appear, as Dickens evidently presumes readers already have a great deal of common sense. When we write for Genesis, we include no metaphor; Dickens's works are full of it.

Directed perception likewise offers great breakthrough-requiring challenges. No vision system can robustly report on what one or two people are doing when what they are doing includes actions such as walk, run, give, take, drop, pick up, dig, bury, approach, follow, flee, chase, hit, catch, push, and lift.

Thus, from the point of view of language and vision, we are better off than when Turing wrote, but surprisingly little better off for 60 years of effort.

Why? I think the answer lies in not understanding the architecture of the brain. We know that vision involves 20 or so identifiable centers that occupy 50% of our cortex. Somehow they all communicate and communicate massively, with massive

amounts of information flowing toward the sensors, not just from the sensors. Language likewise involves many centers connected with one another and even with the centers of vision. Perhaps the needed breakthroughs await our understanding of the exotic architecture in our brains by which multiple centers collaborate, each doing a little on noisy, ambiguous input so that the whole can do a lot.

Turing's final sentence in "Computing Machinery and Intelligence" comes to mind because what he wrote is still true: "We can only see a short distance ahead, but we can see plenty there that needs to be done."

11.8 Turing's Contribution

Fields progress, in part, because they are swept along with better ideas about how to answer fundamental questions. In the 60 years since Turing posed the Turing test as a metric, better ideas have accumulated in computer science, cognitive science, and brain science. Nevertheless, creating a machine with human-like intelligence remains elusive.

Fields also progress when their practitioners formulate better fundamental questions, and my own thinking about how to develop programs that pass Turing's test have evolved into thinking about challenges consequent to the strong story hypothesis and the directed perception hypothesis.

Thus, for me, it is not the test that is at the center of Turing's contribution. It is also not the program of research he proposed. It is, instead, that Turing told us, by defeating counter arguments, that computers can be intelligent. And telling us it can be done unleashed decades of work—some now past, some still in the future—that I am confident will lead, eventually, to programs that are as intelligent as people no matter how big that suitcase word becomes.

References

Awad, H. 2013. *Culturally Based Story Understanding*. Master's thesis, Electrical Engineering and Computer Science Department, MIT.

Chomsky, N. 2005. *Some somple evo devo theses: How True Might they Be for Language?* Talk at the Alice V. and David H. Morris Symposium on Language and Communication, SUNY Stonybrook.

Dreyfus, H.L. 1965. *Alchemy and Artificial Intelligence*. Technical Report P-3244. Rand Corporation.

Havasi, C., J. Pustejovsky, R. Speer, and H. Lieberman. 2009. Digital Intuition: Applying Common Sense Using Dimensionality Reduction. *IEEE Intelligent Systems* 24 (4): 24–35.

Hodges, A. 2012. *Alan Turing: The Enigma, The Centenary Edition*. Princeton: Princeton University Press.

Kurtzweil, R. 2012. Forward. In *The Computer and the Brain*, ed. J. von Neumann, 3rd ed. New Haven: Yale University Press.

Minsky, M. 1961. Steps Towards Artificial Intelligence. In *Computers and Thought*, ed. E.A. Feigenbaum and J. Feldman. Cambridge: MIT Press.

———. 2006. *The Emotion Machine*. New York: Simon and Schuster.

Morris, M.W., and K. Peng. 1994. Culture and Cause: American and Chinese Attributions for Social and Physical Events. *Journal of Personality and Social Psychology* 67 (6): 949–971.

Papert, S. 1968. *The Artificial Intelligence of Hubert L. Dreyfus: A Budget of Fallacies*. Memo 154, Artificial Intelligence Laboratory, MIT.

Tattersall, I. 1998. *Becoming Human*. Orlando: Harcourt.

Turing, A.M. 1950. Computing Machinery and Intelligence. *Mind* 59 (236): 433–460.

Winston, P.H. 2011. The Strong Story Hypothesis and the Directed Perception Hypothesis. In *Technical Report FS-11-01, Papers from the AAAI Fall Symposium*, ed. P. Langley. Menlo Park: AAAI Press.

Winston, P.H. 2012a. The Right Way. *Advances in Cognitive Systems* 1: 23–36.

———. 2012b. The Next 50 Years: A Personal View. *Biologically Inspired Cognitive Architectures* 1: 92–99.

———. 2014. The Genesis Story Understanding and Story Telling System: A 21st Century Step Toward Artificial Intelligence, Memo 019, Center for Brains, Minds, and Machines, MIT. Download at https://cbmm.mit.edu/publications/genesis-story-understanding-and-storytelling-system-21st-century-step-toward.

Chapter 12
From Ockham to Turing – and Back Again

Michael Rescorla

Abstract Beginning with Turing himself, many researchers have suggested that mental processes are Turing-style computations. Proponents typically develop this picture in conjunction with *the formal-syntactic conception of computation* (FSC), which holds that computation manipulates formal syntactic items without regard to their representational or semantic properties. I explore an alternative *semantically permeated* approach, on which many core mental computations are composed from inherently representational elements. The mental symbols over which the computations operate, and hence the computations themselves, have natures inextricably tied to their representational import. We cannot factor out this representational import to generate an explanatorily significant formal syntactic remainder. I argue that the Turing formalism provides no support for FSC over the semantically permeated alternative. I then critique various popular arguments for FSC.

12.1 Computation as Formal Syntactic Manipulation?

Turing (1936/1937) helped launch the computer revolution by advancing the Turing machine as an analysis of symbolic computation. Many authors, including Turing himself (1947, p. 111), have since proposed that the Turing machine or some similar computational formalism might provide a good model for mental activity. This proposal is now sometimes called *the classical computational theory of mind* (CTM). Putnam (1967) introduced philosophers to CTM. Fodor (1975) advanced CTM as a foundation for cognitive science. CTM proved controversial, with many philosophers vigorously dissenting (e.g. Dreyfus 1992; Searle 1980). Researchers have also proposed various rival foundations, including connectionism (Smolensky 1988) and dynamical systems theory (van Gelder 1995). Nevertheless, CTM retains prominent advocates among both philosophers and cognitive scientists.[1]

[1] Ironically, Putnam (1988) has become one of CTM's harshest critics. Fodor (2000, 2008) also now rejects CTM as a theory of cognition *in general*, although he still holds that it well describes many important mental processes (such as perception and language comprehension).

M. Rescorla (✉)
University of California, Los Angeles, CA, USA
e-mail: rescorla@ucla.edu

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_12

I want to discuss how CTM bears upon the traditional picture of the mind as a representational organ. Many notable mental states are *about* a subject matter: my belief *that Donald Trump is president* is about Donald Trump; my desire *that I drink some water* is about water; my perceptual experience *as of a red sphere standing before me* is about redness and sphericality; and so on. Historically, most philosophers have assigned a crucial role to "aboutness" (or *intentionality*) when elucidating reasoning, decision-making, perception, and other paradigmatic mental activities. Hence the proliferation of theories trafficking in reference, truth-conditions, representational content, propositions, etc. All these theories prioritize *intentional descriptions*, which individuate mental states at least partly through their representational or semantic properties.

What is the relation between computational modeling and intentional description?[2]

According to current orthodoxy, computation manipulates formal syntactic items without regard to their representational or semantic properties. I will call this *the formal-syntactic conception of computation* (FSC). Fodor (1981, pp. 226–227) offers a classic statement: computational processes "are formal because they apply to representations in virtue of (roughly) the *syntax* of the representations… What makes syntactic operations a species of formal operations is that being syntactic is a way of *not* being semantic. Formal operations are the ones that are specified without reference to such semantic properties as, for example, truth, reference, and meaning." Other proponents of FSC include Field (2001), Gallistel and King (2009), Haugeland (1985), Pylyshyn (1984), and Stich (1983). All these authors combine FSC with CTM. According to CTM + FSC, mental activity manipulates formal syntactic items without regard to their representational or semantic properties. Perhaps mental states *have* representational properties. But we should delineate *non-intentional syntactic descriptions* that leave those properties unmentioned.

Stich (1983) espouses an extreme version of the formal syntactic approach. He advises cognitive science to describe the mind through purely syntactic models that ignore representational import. He recommends that scientific psychology jettison mental content altogether. Few proponents of CTM + FSC condone this extreme rejection of mental content. Most proponents try to secure a central explanatory role for formal mental syntax while *also* preserving a central role for representation (Fodor1987, 2008). All proponents agree that cognitive science should include a level of description that characterizes mental states in syntactic, non-semantic terms.

I think that FSC fits many computations quite well, including computations executed by standard personal computers. However, I reject FSC as a theory of computation *in general*. There is no sound reason to hold that all computation

[2] In addressing this question, I restrict attention to the Turing machine and kindred computational formalisms. I do not consider computation by neural networks, because I am concerned solely with *classical* versions of the doctrine that the mind is a computing system. For purposes of this paper, "computation" means "Turing-style computation." See (Gallistel and King 2009) for a recent, detailed case that Turing-style models of the mind offer important advantages over neural network models.

manipulates formal syntactic items. (Rescorla 2012b) introduces an alternative *semantically permeated* conception that integrates representation much more thoroughly into computational modeling. On the semantically permeated conception, certain computational models individuate computational states through their representational properties *as opposed to* any alleged formal syntactic properties. Specifically, computational models of mental activity can type-identify mental states in representational terms rather than formal syntactic terms. We can model the mind computationally without postulating formal mental syntax.[3]

In Sect. 12.2, I present basic elements of the semantically permeated conception. In Sect. 12.3, I argue that the Turing machineformalism is quite hospitable to semantically permeated computation. In Sect. 12.4, I discuss how current explanatory practice within cognitive science bears upon the contrast between formal-syntactic computation and semantically permeated computation. In Sect. 12.5, I critique some popular philosophical arguments for FSC. I will not argue that the semantically permeated conception is superior to FSC. Nor will I develop the semantically permeated conception in full detail. My discussion is programmatic. I seek only to convince you that current philosophical discussion has precipitously embraced a formal-syntactic picture of mental computation, at the expense of an equally appealing semantically permeated alternative.

12.2 Individuating Mental States

Let us begin by considering Fodor's version of CTM. Fodor advocates *the representational theory of mind* (RTM), which postulates *mental representations* comprising *the language of thought* (or *Mentalese*). Mentalese contains primitive symbols and compounding devices for generating complex expressions. It has a *compositional semantics*: the meaning of a complex Mentalese expression is determined by the meanings of its parts and the way those parts are combined. As Fodor (1987) emphasizes, RTM explains two crucial phenomena: *systematicity* (there are systematic relations among which thoughts a thinker can entertain) and *productivity* (even though the mind is finite, one can entertain a potential infinity of thoughts). For instance, we explain productivity by positing a finite base of primitive Mentalese symbols, combinable through compounding devices into complex expressions. Iterated application of the compounding devices generates a potential infinity of expressions.

According to CTM + RTM, mental activity instantiates Turing-style computation over the language of thought. Mental computation stores Mentalese expressions in memory locations, manipulating those expressions in accord with mechanical rules. To delineate a computational model of a mental process, we specify the Mentalese expressions manipulated by the process, and we isolate mechanical rules governing how the process manipulates those expressions.

[3] Burge (2010, pp. 95–101) and Peacocke (1994) propose somewhat similar treatments of computation. For critical discussion of these and other neighboring positions, see (Rescorla 2012a).

I assume that CTM + RTM is correct. I focus on the following key question: when we construct computational models of mental activity, to what extent should representational properties inform how we individuate Mentalese expressions?

12.2.1 Semantic Neutrality, Indeterminacy, and Permeation

According to Fodor, "mental representations… have both formal and semantic properties," and "mental representations have their causal roles in virtue of the formal properties" (1981, p. 26). Crucially, formal properties underdetermine semantic properties: "mental representations can differ in content without differing in their intrinsic, formal, nonrelational, nonsemantic properties" (1991, p. 298). To illustrate, consider Putnam's (1975) Twin Earth thought experiment. Oscar's mental states represent water, while Twin Oscar's corresponding mental states represent twater (the substance on Twin Earth). This is a *semantic* difference between Oscar's mental states and Twin Oscar's corresponding mental states. But Fodor holds that Oscar's mental representations have the same *formal syntactic* properties as Twin Oscar's corresponding mental representations.

On Fodor's approach, computational models of the mind should type-identify Mentalese symbols through their formal syntactic properties rather than their semantic properties. Phrased in more ontologically loaded terms, Fodor postulates an array of *formal syntactic types* to serve as the items manipulated by mental computation. For example, he postulates a formal syntactic type WATER that could denote either water or twater, depending on the thinker's causal relations to the external world. Initially, Fodor (1981, pp. 225–253) held that formal syntactic type *constrains* meaning while leaving meaning underdetermined. WATER could denote water or twater, but it could not denote dogs. Fodor's later work (1994, 2008) suggests a stronger indeterminacy thesis: a Mentalese syntactic type could have had an *arbitrarily* different meaning, had it figured differently in the thinker's psychology or her causal interactions with the world. WATER could denote dogs, or Bill Clinton, or anything else. Many researchers explicitly endorse the stronger indeterminacy thesis (Egan 1992, p. 446), (Field 2001, p. 58), (Harnad 1994, p. 386), (Haugeland 1985, p. 91, pp. 117–123), (Pylyshyn 1984, p. 50).

I say that an entity is *semantically indeterminate* when it does not have its meaning essentially. A semantically indeterminate entity could have had a different meaning without any change in its fundamental nature, identity, or essence. I say that an entity is *semantically neutral* when it bears an arbitrary relation to its meaning (assuming it even has meaning). A semantically neutral entity could have had *arbitrarily different* meaning, or no meaning at all, without any change in its fundamental nature, identity, or essence. Semantic neutrality entails semantic indeterminacy, but not vice-versa: semantic indeterminacy entails only that the entity could have had *some* different meaning, while semantic neutrality entails that the entity could have had *any* different meaning. Egan, Field, and Haugeland hold that Mentalese syntactic types are semantically neutral, as do most other

contemporary advocates of CTM. Fodor's early work treats Mentalese syntactic types as semantically indeterminate, while his later work seems to treat them as semantically neutral.

I will explore computational models that type-identify Mentalese symbols at least partly through their semantic properties. In this spirit, I say that an entity is *semantically permeated* when we cannot change its meaning while holding fixed its fundamental identity, nature, or essence. A semantically permeated symbol is not a piece of formal syntax requiring an interpretation. Rather, the semantics of the symbol is built into the symbol's inherent nature. The symbol "comes with its meaning attached." I propose that we postulate an array of semantically permeatedMentalese symbols (or mental representations). For example, we can posit a Mentalese word DOG that necessarily denotes dogs, a Mentalese word SQUARE that necessarily denotes the property of being square, a Mentalese word AND that necessarily expresses conjunction, and so on.

To understand my proposal, one must keep the type/token distinction firmly in mind. I propose that we postulate semantically permeated Mentalese symbol-*types*, not that we postulate semantically permeated *tokens*. If we postulate a Mentalese symbol-type DOG that has its meaning essentially, we do not thereby claim that tokens of this type have their meanings essentially. Any possible token of DOG denotes dogs *so long as it is a token of DOG*. A given token of DOG might not have denoted dogs, but then it would not have been a token of DOG.

Although semantically permeated Mentalese individuation departs significantly from contemporary orthodoxy, it has a strong historical basis. Going back all the way to the fourteenth century, consider William of Ockham. Like most philosophers, Ockham holds that the connection between a natural language word and what it represents is arbitrary, since we can change the word's meaning as we please by changing our linguistic conventions. The English word "dog" could just as easily have denoted cats. Ockham also postulates a mental language whose elements have fixed, unchangeable denotations. He describes the contrast between natural language and mental language as follows (*Summa Logicae*, I.1):

> A concept or mental impression signifies naturally whatever it does signify; a spoken or written term, on the other hand, does not signify anything except by free convention. From this follows another difference. We can change the designation of the spoken or written term at will, but the designation of the conceptual term is not to be changed at anybody's will.

In my terminology, Ockham holds that natural language words are semantically neutral but that Mentalese words are semantically permeated. Ockham offers no hint that we can "hive off" the denotation of a Mentalese word, leaving behind a non-semantic syntactic residue. Whereas Fodor posits formal mental syntax subject to varying interpretations, Ockham posits a mental language whose nature fixes a unique interpretation.

There are many other precedents for my proposal. As Burge notes, a traditional view holds that "concepts' identities are inseparable from their specific intentional properties or functions. Thus a concept of an eclipse could not be the concept that it

is if it did not represent, or if it were not about, eclipses" (2007, p. 292). In my terminology, the traditional view holds that concepts are semantically permeated. Rather than speak of "concepts," I speak of "Mentalese symbols" or "mental representations." I choose these locutions so as to accommodate *non-conceptual representations* (if such there be). For example, Burge thinks that perception is non-conceptual. He postulates *perceptual attributives*, which are perceptual analogs to predicates or concepts. He holds that "perceptual attributives determine, or *specify*, the attributes they attribute. They are not only *as of* the attributes; they are as of the same attribute in every context of use and with regard to any possible situation" (2010, p. 76). In my terminology, Burge holds that perceptual attributives are semantically permeated. My formulations are general enough to encompass perceptual attributives and other putatively non-conceptual mental representations.

12.2.2 Developing the Semantically Permeated Viewpoint

I characterized "semantically permeation" in fairly sketchy terms. In particular, I used the rather nebulous term "meaning," rather than some more precise technical term. One might develop the semantically permeated viewpoint in various directions, depending on how one glosses the crucial term "meaning." For example, one might type-identify Mentalese expressions by citing Russellian propositions, or sets of possible worlds, or Fregean senses.[4] There are many other options. A complete semantically permeated theory must choose among these options. I remain neutral regarding such details, which will not affect my argumentation. For present purposes, what matters is that a semantically permeated individuative scheme classifies mental states *at least partly* through their representational import.[5]

The Twin Earth thought experiment shows that representational import does not always supervene upon internal neurophysiology. Thus, semantic permeation induces an *externalist* approach to Mentalese symbol individuation. For example, semantically permeated theorists will postulate a Mentalese word WATER that necessarily denotes water. When Oscar wants to drink water, he stands in a certain

[4] For representative modern treatments of the Russellian, possible worlds, and Fregean approaches, see (Salmon 1986), (Stalnaker 1984), and (Peacocke 1992) respectively. In (Rescorla 2012b), I develop a broadly Fregean version of semantically permeated CTM + RTM. Schneider (2011, p. 100) rejects a semantically permeated individuative scheme for Mentalese, partly because she thinks that such a scheme cannot handle *Frege cases*, i.e. cases where a thinker represents the same entity under different modes of presentation. Schneider mistakenly assumes that a semantically permeated scheme must type-identify mental symbols in Russellian fashion. She does not even consider an alternative Fregean approach that type-identifies mental symbols by citing externalistically individuated modes of presentation.

[5] Subtle issues arise concerning the "compounding devices" that generate complex Mentalese expressions. To ensure that complex Mentalese expressions are semantically permeated, we must isolate compounding devices with fixed compositional import. For preliminary discussion, see (Rescorla 2012b). This paper focuses on issues raised by the individuation of primitive Mentalese words.

cognitive relation to the Mentalese word WATER. Twin Oscar's mental states do not represent water, so he does not stand in any significant cognitive relation to WATER. Instead, he stands in a cognitive relation to a type-distinct Mentalese word TWATER. Even though Oscar and Twin Oscar are neurophysiological duplicates, they entertain different Mentalese word-types. Externally determined denotation plays a crucial role in type-identifying the relevant words. Oscar and Twin Oscar have different mental languages.

Some philosophers respond to the Twin Earth thought experiment by recommending that we replace *wide content* (which does not supervene upon internal neurophysiology) with *narrow content* (which does so supervene). On Fodor's (1981, p. 227, p. 240) early view, each formal syntactic type determines a unique narrow content but not a unique wide content. So formal syntactic type underdetermines vital aspects of a mental state's representational import (e.g. WATER might denote either water or twater). I count these Fodorian formal syntactic types as semantically indeterminate, even though each such type allegedly has a certain kind of content (narrow content) essentially. Semantic permeation requires that an entity have its meaning essentially *in some sense of "meaning" correlative with "representational import."*[6]

Phenomena such as context-sensitivity and reference failure might lead one to refine, qualify, or weaken the semantically permeated viewpoint in various ways. There are many complexities here that I must skirt.[7] This paper emphasizes abstract issues that arise however exactly one develops the semantically permeated viewpoint.

12.2.3 Explanation and Taxonomization

Critics often lambast semantically permeated entities as mysterious or obscure. Putnam complains that "[n]one of the methods of representation we know about has the property that the representations *intrinsically* refer to whatever it is that they are used to refer to" (1988, p. 21). He warns us to be "highly suspicious of theories that

[6] Around the mid-1990s, Fodor abandons narrow content. A constant element in his position is his emphasis upon formal syntactic types that underdetermine representational import. Aydede (2005) proposes that we type-identify Mentalese symbols partly by what he calls their "semantic properties." However, he seems reluctant to develop this proposal in an externalist direction (p. 203, fn. 26). He instead inclines towards permeation by some kind of narrow content. Ultimately, then, Aydede's proposal seems closer to Fodor's (1981) view than to my view.

[7] For example, one might postulate a Mentalese demonstrative THAT that can only denote some demonstrated entity but whose *particular* denotation depends upon context. More specifically, one might propose that THAT does not have its *denotation* essentially but does have something like its *character* in the sense of (Kaplan 1989) essentially. This proposal individuates THAT partly by *context-insensitive* aspects of its representational import but not by *context-sensitive* aspects. Does the proposal classify THAT as semantically permeated? To answer the question, one requires a more precise definition of "semantic permeation" than I have provided.

postulate a realm of 'representations' with *such* unlikely properties" (p. 22). He hints, without asserting, that positing such representations is tantamount to positing entities with "magical" powers.

I find Putnam's objection unconvincing. A key point here is that semantically permeated types are *types*, whose primary role in our discourse is to taxonomize tokens. Any psychological theory must adopt a taxonomic scheme for categorizing token mental states, processes, and events. We reify the categories by positing a collection of types. The types are abstract entities corresponding to our classificatory procedures. Semantically indeterminate types correspond to a taxonomic scheme that underdetermines meaning. Semantically neutral types correspond to a taxonomic scheme that leaves meaning completely unconstrained. Semantically permeated types correspond to a taxonomic scheme that takes meaning into account. As Burge (2007, p. 302) notes, there is nothing obscure or magical about the latter taxonomic scheme. On the contrary, it figures centrally in ordinary mentalistic discourse. Setting aside generalized skepticism about abstract entities, I see no *metaphysical* problem about semantically permeated Mentalese types. For instance, there is nothing supernatural about a Mentalese symbol-type DOG that refers to dogs by its essential nature. Semantically permeated Mentalese symbol-types are mere "ontological correlates" to a taxonomic scheme that type-identifies mental states, events, or processes partly through their semantic properties.

Individuation serves explanation. When we debate the proper individuation of Mentalese symbols, we are ultimately debating the proper format for psychological explanation. The central issue here is not ontological but explanatory. How do our best psychological explanations type-identify mental states, events, and processes? Should we employ a taxonomic scheme that cites representational properties of mental states? Or should we employ a taxonomic scheme that leaves representational properties underdetermined?

One might find room in one's theorizing for *both* kinds of taxonomic scheme. For example, Fodor (1998) devotes an entire book to entities that he calls "concepts." A Fodorian concept determines a unique denotation (1998, pp. 20–21), (2008, p. 70), so it is semantically permeated. Thus, Fodor postulates both semantically permeated types (concepts) *and* semantically indeterminate types (formal syntax).

12.2.4 Formal-Syntactic Computation Versus Semantically Permeated Computation

CTM+FSC embraces formal syntactic taxonomization. Proponents view the mind as a Turing-style device that manipulates semantically indeterminate syntactic types. Mental computation operates over these types, without regard to representational properties. In that sense, the mind is a "syntactic engine."

One must distinguish between *syntactic* description and *neurophysiological* description. Beginning with Putnam (1967), advocates of CTM have repeatedly stressed that computational models are *multiply realizable*: systems with wildly heterogeneous physical properties can instantiate the same computational properties. In particular, formal syntactic description is supposed to be far more abstract than neurophysiological description, citing properties shareable by diverse physical systems: carbon-based, silicon-based, and so on. Fodor (2008, p. 91), Gallistel and King (2009, pp. 137–148), Haugeland (1985, p. 5), Stich (1983, p. 151), and virtually all other proponents of CTM + FSC tout multiple realizability as a prime benefit of their approach. They commend syntactic description for ignoring incidental neurophysiological details. Thus, CTM + FSC prioritizes formal syntactic descriptions that abstract away from both semantic properties *and* neurophysiological properties. Proponents hold that mental computation operates over semantically indeterminate, multiply realizable syntactic types.

On the alternative picture that I will explore, many notable mental computations operate over semantically permeated Mentalese expressions without intercession by formal mental syntax. These computations store semantically permeated Mentalese symbols in memory locations, manipulating the symbols according to mechanical rules. The symbols lack formal syntactic properties of any theoretical interest. On this picture, many core mental computations are composed from inherently representational building blocks. The mental symbols over which the computations operate, and hence the computations themselves, have natures inextricably tied to their representational import. We cannot factor out representational import to generate an explanatorily significant formal syntactic remainder.

The alternative picture has a positive component and a negative component. The positive component is that we promote good psychological explanation of various core mental phenomena by delineating Turing-style models defined over semantically permeated Mentalese symbols. The negative component is that formal syntactic taxonomization adds no explanatory value to theorizing about those same phenomena. I assume that fruitful explanation is our best guide to underlying natures. Given this assumption, the positive and negative components jointly indicate that certain core mental computations manipulate mental symbols whose natures are inextricably tied to their representational import.

A satisfying scientific psychology will certainly include *non-representational neurophysiological* descriptions. But will it include *non-representational syntactic* descriptions? Given multiple realizability, neurophysiological description is fundamentally distinct from formal syntactic description. If we allow a sufficiently disjunctive or gerrymandered taxonomic scheme, it may be that every mental representation has a semantically indeterminate, multiply realizable syntactic type. What I doubt is that each mental representation has an *explanatorily significant* semantically indeterminate, multiply realizable syntactic type. There are indefinitely many ways to type-identify mental states. Most taxonomic schemes hold no interest for us (e.g. a scheme that type-identifies mental states by citing the current temperature on Mars). Only certain special taxonomic schemes serve our explanatory ends. Does a formal syntactic taxonomic scheme for mental representations serve psychological

explanation? Or is formal mental syntax a gratuitous theoretical posit? The issue here is not whether we *can* individuate mental representations in formal syntactic fashion. The issue is whether we *should*.

12.3 Turing Computation Over Semantically Permeated Types

Philosophers and cognitive scientists often suggest that formal syntax is an integral component of computational modeling. In this spirit, Fodor (1981, p. 241) writes: "Computations just *are* processes in which representations have their causal consequences in virtue of their form." Similarly, Gallistel and King (2009, p. 107) say that the symbols manipulated by Turing machines "are to be regarded as purely arbitrary symbols (really data), having no more intrinsic reference than magnetic patterns," while Haugeland (1985) defines a computer as "a *symbol-manipulating machine*" (p. 106), where "the meanings of symbols (e.g. words) are *arbitrary*… in the sense that there is no intrinsic reason for them to be one way rather than another" (p. 91). These passages, and many similar passages found throughout the literature, are offered as uncontroversial remarks that everyone should accept.

I believe that philosophical discussion of computation should ground itself in the mathematical theory of computation, construed broadly to include recursion theory, complexity theory, and theoretical computer science. If we take mathematical computation theory as our guide, then there is no reason to insist that computational modeling requires formal syntax. Abstract mathematical models of computation are indifferent between semantic neutrality, indeterminacy, and permeation. Nothing about the *mathematics* of computation mandates explanatorily significant formal syntactic types.

To see why, let us examine the Turing machineformalism. Philosophers commonly describe Turing machines along the following lines:

> **(1)** A Turing machine consists of a scanner and infinite paper tape, divided into cells. The scanner manipulates strings of strokes inscribed upon the tape. The scanner can erase a stroke, write a stroke, or move to the left or right. A machine table enshrines routine, determinist rules governing these manipulations. The scanner's action is determined entirely by its current "internal state" and by the tape's configuration at the scanner's current location.

A string of stokes does not have essential meaning or content. We could interpret it however we please. Thus, (1) enshrines the orthodox conception of computation as defined over semantically indeterminate entities.

Heuristic descriptions such as (1) have undeniable pedagogical value. However, they do not capture the notion of Turing machine in full generality. Turing machines can operate over entities other than strings of strokes. A proper description should invoke some general notion of "symbol," rather than restricting attention to strokes. Another problem with (1) is its talk about a "scanner" that "moves" along a "tape."

Although Turing indulges in such talk, we can only construe it metaphorically. Why restrict attention to literal tapes rather than other physical embodiments, such as silicon chips? Finally, how can a rigorous mathematical theory deploy informal notions such as "motion"?

A proper formulation must divest (1) of all picturesque embellishments. Once we jettison inessential metaphors, the following core notion emerges:

> **(2)** A Turing machine contains a central processor and an infinite linear array of memory locations. The scanner can access one memory location at each stage of computation. There is a finite set of primitive symbols, any of which can be inscribed at a memory location. The processor can erase a symbol from the currently accessed memory location, write a symbol to the currently accessed memory location, or access the previous or next location in the memory array. A machine table enshrines routine, deterministic rules governing these manipulations. The central processor's action is determined entirely by its current "internal state" and by the contents of the currently accessed memory location.

I submit that (2) provides a much more accurate characterization of Turing computation than (1).

Once formulated with proper generality, Turing's conception enshrines no bias towards semantic indeterminacy. A semantically permeated symbol can be inscribed or erased at a memory location. Routine deterministic instructions can mention semantically permeated symbols. Thus, there is no obvious bar to a Turing machine defined over semantically permeated symbols. Such a machine contains an infinite, linearly structured array of memory locations (the "cells" of the "tape"). It contains a central processor (the "scanner"), which can access one memory location at a time. It performs the same elementary operations as a Turing machine defined over semantically indeterminate items: accessing the next memory location in the linear memory array ("moving to the right"); accessing the previous memory location in the linear memory array ("moving to left"); inscribing a symbol in a memory location; erasing a symbol from a memory location. It merely performs these operations upon semantically permeated rather than semantically indeterminate items.

Somewhat more formally, consider how modern computation theory codifies Turing's conception in mathematically rigorous fashion. Taylor's definition (1998, p. 73), which is representative of the modern literature, runs roughly as follows:

> **(3)** A Turing machine M is an ordered quadruple $<Q, \Sigma, q_{init}, \delta>$, where Q is a nonempty finite set (the set of *states* of M); Σ is a nonempty finite *alphabet of symbols*, including the null symbol (*blank space*); $q_{init} \in Q$ is a privileged *initial state*; and δ is a partial function (the *transition function* of M) from $\Sigma \times Q$ to $(\Sigma \cup \{L, R\}) \times Q$.

(3) mentions a finite alphabet of symbols Σ but say nothing about the nature of those symbols. As far as the definition goes, the symbols could be semantically neutral, semantically indeterminate, or semantically permeated. So the abstract modern definition of Turing machine is entirely congenial to a semantically permeated individuative scheme for Mentalese, including a scheme that takes wide content into account. The abstract definition provides no reason to associate each Mentalese word with a formal syntactic type.

To illustrate, suppose that the transition function δ maps $<r_1, q_1>$ to $<r_2, q_2>$, where $r_1, r_2 \in \Sigma$ and $q_1, q_2 \in Q$. This corresponds to the following mechanical rule:

> **(R)** If the central processor is in state q_1, and if it accesses a memory location that contains symbol r_1, then replace r_1 with r_2 and shift to central processor state q_2.

Neither rule R nor our formalization through δ requires that r_1 and r_2 be semantically indeterminate. Rule R applies just as well to symbols individuated through their representational properties as to symbols individuated through formal syntactic properties. For example, r_1 might be a Mentalese word WATER that necessarily denotes water.

We should reject the popular philosophical view that all Turing machines are defined over semantically indeterminate items. This popular view is tempting only if one confines attention to overly picturesque descriptions such as (1). The rigorous, abstract notion of Turing machine, as encapsulated by (2) and formalized by (3), generates no impetus towards formal syntactic computational vehicles.[8] We can therefore model mental activity as Turing-style computation over semantically permeated Mentalese types, without postulating formal mental syntax. Genuinely computational models can describe mechanical manipulation of mental representations *individuated partly through their representational import*. Despite what many researchers suggest, the mathematical study of computation does not favor FSC.

Some readers may worry that semantically permeated computation requires an "inner homunculus" who interprets Mentalese expressions. Consider Rule R, and suppose that the types r_1 and r_2 are semantically permeated. For mental computation to conform reliably to R, surely it must evaluate whether a given token has type r_1. And surely that requires evaluating whether the token has appropriate semantic properties. More specifically, suppose r_1 is a Mentalese word WATER that necessarily denotes water. In manipulating this word, doesn't the system first need to check whether the word denotes water versus twater? Wouldn't that require deciding whether the system is located on Earth versus Twin Earth?

These worries are misplaced. A computational model defined over Mentalese expressions taxonomizes mental states and operations by citing mental symbol-types. In any normal case, the symbol-types are not *objects* of cognition or awareness. Rather, they are abstract types that *we theorists* cite to taxonomize mental states, events, and processes. Mental computation does not normally represent mental symbols. It *tokens* the symbols. Conformity to Rule R does not require that a computational system evaluate whether tokens have types r_1 or r_2. Conformity simply requires that the system move appropriately between mental states that have types r_1 and r_2. In the special case where r_1 is semantically permeated, the system need not evaluate representational properties of mental states, because it need not evaluate whether states have type r_1. It need merely move appropriately between states with appropriate representational properties. For example, suppose r_1 is a

[8] In (Rescorla 2012b), I offer a similar diagnosis for other mathematical models of computation, including the *register machine* and the *lambda calculus*. In each case, I argue that the relevant computational formalism is hospitable to semantically permeated individuation.

Mentalese word WATER that necessarily denotes water. In manipulating this word, the system need not evaluate whether tokens represents water versus twater, or whether the system's location is Earth or Twin Earth. The system need merely transit in the right way among mental states, some of which represent water. As long as the system so transits, it conforms to rule R.

Implementing a semantically permeated Turing-style model does not require *evaluating* representational properties of mental states. It requires *transiting* appropriately between mental states with appropriate representational properties. Thus, there is no reason to suspect that semantically permeated computation requires an inner homunculus.

12.4 Modeling Mental Computation

I have presented two opposing conceptions of computation: the formal-syntactic conception and the semantically permeated conception. How do the two conceptions apply to computation in physical systems? In my view, the answer depends upon the physical system. Computer science routinely offers semantically indeterminate models of artificial computing systems (e.g. personal computers). The explanatory and pragmatic success of these models provides strong evidence that the relevant systems compute by manipulating formal syntactic items. However, we cannot immediately infer that *minds* likewise compute by manipulating formal syntactic items. There are many differences between minds and artificial computing systems. Our best models of the former may differ markedly from our best models of the latter.

In Sect. 12.2.4, I suggested that certain core mental phenomena are best handled through semantically permeated rather than semantically indeterminate computational modeling. I intend this suggestion in a tentative, conjectural spirit. I also allow that some important mental phenomena are best described through semantically indeterminate computational models, or through models that contain a mixture of semantically indeterminate and semantically permeated symbols.[9] At present, current cognitive science does not offer anything resembling well-confirmed Turing-style models of specific mental phenomena. As well-confirmed Turing-style models emerge, we can assess the extent to which they are semantically neutral, indeterminate, or permeated. Until then, we should keep all theoretical options open.

Fodor (1975, 1981) and his allies frequently claim that cognitive science postulates mental computation over semantically indeterminate formal syntactic types. They claim that formal syntactic description figures essentially within our best

[9] For example, Gallistel and King (2009) mount a compelling case that *dead reckoning* manipulates symbols inscribed in read/write memory. (Rescorla 2013b) suggests that current science describes certain cases of invertebrate dead reckoning in non-representational terms. So these may be cases where formal-syntactic computational description is more apt than semantically permeated computational description.

scientific theories of mental activity. I think that such claims vastly overstate the centrality of formal mental syntax to contemporary scientific practice. Although current cognitive science may describe *certain* mental phenomena in formal syntactic terms, it eschews formal syntactic description when explaining numerous core mental phenomena. It describes numerous mental processes in representational terms *as opposed to* formal syntactic terms. I will not defend my assessment here. But I will illustrate by considering one important mental process: *perception*.

The perceptual system reliably estimates distal properties (e.g shapes, sizes, and distances) based upon proximal sensory stimulations (e.g. retinal stimulations). For example, the visual system estimates the distance of a perceived body based upon numerous visual cues, including convergence, binocular disparity, linear perspective, motion parallax, and so on (Palmer 1999, pp. 200–253). The visual system also consults distance cues when estimating distal size: if bodies *A* and *B* subtend the same retinal angle but distance cues indicate that *A* is farther away, then the perceptual system will typically estimate that *A* is larger than *B* (Palmer 1999, pp. 314–327). Perceptual psychology studies such phenomena. It provides detailed psychological models that explain how the perceptual system estimates distal properties based upon proximal sensory stimulations (Feldman 2015), (Knill and Richards 1996), (Vilares and Körding 2011).

As Burge (2010) and Peacocke (1994) emphasize, the science routinely describes perceptual activity in representational terms. It individuates perceptual states through representational relations to specific shapes, sizes, distances, and so on. For example, models of distance perception type-identify perceptual states as estimates of specific distances, while models of size perception type-identify perceptual states as estimates of specific sizes. Perceptual psychology does *not* attribute formal syntactic properties to perceptual states. As Burge puts it, "there is no explanatory level in the actual science at which any states are described as purely or primitively syntactical, or purely or primitively formal. One will search textbooks and articles in perceptual psychology in vain to find mention of purely syntactical structures" (2010, p. 96). Taking perceptual psychology as our guide, it is sheer fantasy to postulate that perceptual activity manipulates formal syntactic items. The science describes how proximal sensory input *as characterized neurophysiologically* determines a perceptual estimate *as characterized representationally*. Formal mental syntax plays no role (Rescorla 2015).

Any adequate theory of perception must, among other things, illuminate the *neural mechanisms* that underlie perceptual estimation. Perceptual psychology offers some theories in this vein (Pouget et al. 2013). The theories tend to be rather more speculative than theories couched solely at the representational level. However, we can all agree that a completed perceptual psychology will contain non-representational *neural* descriptions. The question is whether it should include non-representational *syntactic* descriptions. Should we postulate multiply realizable, semantically indeterminate types manipulated during perceptual processing? Current science postulates no such types. It describes perceptual activity

in *representational* terms and *neurophysiological* terms, but never in *formal syntactic* terms.

One might recommend that we *supplement* perceptual psychology with formal syntactic descriptions. But that recommendation needs sound backing. Nothing about current perceptual psychology taken in itself suggests any significant explanatory role for formal syntactic computational modeling.

Beginning with Quine (1960), some philosophers have insisted that intentionality should not figure in mature scientific psychology. They argue that intentional discourse is unclear, interest-relative, explanatorily bankrupt, or otherwise unscientific (Churchland 1981), (Field 2001), (Stich 1983). This attitude prompts Stich (1983) to recommend that we replace intentional psychology with purely syntactic computational modeling of the mind. Even philosophers more sympathetic to intentionality, such as Fodor (1987, pp. 16–26), often suggest that formal syntactic modeling provides intentional psychology with a secure scientific grounding that it would otherwise lack.

I dispute all these assessments. There is nothing unscientific about intentional discourse per se. If we consider how current science actually works, rather than how various philosophers think that it should work, then representation looks like a thoroughly legitimate theoretical construct. In particular, it has repeatedly proved its explanatory value within perceptual psychology. The diverse arguments offered by Churchland, Field, Quine, and Stich against the scientific credentials of intentional discourse are notably less compelling than the intentional explanations routinely offered by perceptual psychology. Moreover, I see no clear respect in which formal syntactic modeling is more scientifically respectable than intentional psychology. Representation rather than formal mental syntax occupies center stage within our current best scientific theories of perception.

Over the past few decades, philosophers have defended FSC through various arguments that formal syntactic description offers decisive advantages over representational description. I cannot rebut all these arguments here. But I will now critique several of the most prominent arguments.

12.5 The Mechanisms of Cognition

A natural thought is that semantically permeated computational description does not seem *mechanical* enough. Shouldn't a truly mechanical account "bottom out" in mindless responses to formal syntactic items? Consider rule R from §3, and suppose again that the relevant types r_1 and r_2 are semantically permeated. Some readers will insist that a physical system conforms to R only by virtue of conforming to the formal syntactic rule:

(R*) If the central processor is in state q_1, and if it accesses a memory location that contains symbol r_1*, then replace r_1* with r_2* and shift to central processor state q_2,

where r_1* and r_2* are formal syntactic types. Even if intentional description is useful for certain purposes, formal syntax plays an essential causal or explanatory role.

The argument as stated is highly schematic. Let us consider how one might develop it less schematically.

12.5.1 The Causal Relevance of Content

An extreme version of the argument maintains that representational properties are *causally irrelevant* to mental activity. Suppose we grant this premise. Then it becomes natural to pursue semantically indeterminate models, so as to secure causal theories of mental computation. Egan (2003), Gallistel and King (2009), and Haugeland (1985) argue along these lines.

Luckily, there are good reasons to reject the crucial premise that representational properties are causally irrelevant. These issues have been thoroughly litigated over the past few decades, so I will just briefly highlight four key points:

(1) We have a strong pre-theoretic intuition that mental content is causally relevant. For example, whether I want to drink water versus orange juice seems causally relevant to whether I walk to the sink versus the refrigerator. To deny pre-theoretic intuitions along these lines is to embrace radical epiphenomenalism.
(2) As emphasized in Sect. 12.5, cognitive science offers numerous explanations that type-identify mental states in representational terms. The explanations certainly look like causal explanations (Burge 2007, pp. 344–362). For example, current theories of sensorimotor control describe how an intention to move my finger to a certain egocentric location causes certain muscle activations (Bays and Wolpert 2007). Taken at face value, these theories depict how representational properties of my intention causally influence my muscle activations.
(3) In (Rescorla 2014a), I provide a detailed argument that mental content – including wide content – is causally relevant to mental computation. In particular, I argue that representational properties can causally influence elementary computational operations. Thus, genuinely causal explanatory generalizations can individuate computational states representationally.
(4) Various widely circulated arguments maintain that mental content – especially wide content – is causally irrelevant to mental activity. I agree with Burge (2007, pp. 316–382) that these arguments are seriously flawed. I critique some of them in (Rescorla 2014a).

Given (1)–(4), we may with clear conscience pursue intentional causal explanations of mental activity. Semantically permeated versions of CTM instantiate this explanatory strategy.

I now elaborate upon (4) by critiquing two representative arguments that mental content is causally irrelevant.

Haugeland observes that "meanings (whatever exactly they are) don't exert physical forces" (1985, p. 39). If we say that meanings "affect the operation of the mechanism," then "all the old embarrassments return about exerting forces without having any mass, electric charge, etc.: meanings as such simply cannot affect a physical mechanism" (p. 40). He concludes that "meanings don't matter" to computational operations (p. 44). On that basis, he urges us to model mental activity as formal manipulation of semantically neutral symbols.

I agree with Haugeland that meanings do not exert physical forces. Meanings are abstract entities, so they do not participate in causal interactions. It does not follow that intentional properties are causally irrelevant. Intentional *properties* can be causally relevant even though intentional *contents* do not enter into causal transactions. To adopt a well-worn analogy: numbers are abstract entities, so they cannot causally interact with physical objects; yet an object's mass, as measured by some number, is causally relevant to physical processes. One can specify causally relevant properties by citing abstract entities. Thus, Haugeland's argument does not establish that genuinely causal explanations should ignore intentional content.

Haugeland offers a second argument that intentional properties do not "matter" to mental computation. He suggests that meanings "matter" only if there is an inner homunculus capable of "'reading' the symbols in [the thinker's] mind, figuring out what they mean, looking up rules of reason, deciding which ones to apply, and then applying them correctly" (p. 41). He denies that such a homunculus exists. In a similar vein, Gallistel and King write: "[f]or many purposes, we need not consider what the symbols refer to, because they have no effect on how a Turing machine operates. The machine does not know what messages the symbols it is reading and writing designate (refer to)" (2009, p. 108).

I agree that no inner homunculus interprets Mentalese symbols. I agree that a typical Turing machine does not "know" the semantics of symbols it manipulates. We should not infer that representational properties do not "matter" or that they have "no effect" on computation. If I throw a baseball at a window, then the window does not contain a homunculus that inspects baseball momentum. The window does not know the baseball's momentum. Nevertheless, the baseball's momentum is causally relevant to whether the window breaks. The momentum "matters" to the window's breaking. Likewise, representational properties can "matter" to mental computation even though no inner homunculus inspects a symbol's meaning. There is no cogent inference to the causal irrelevance of representational properties from the non-existence of an inner homunculus, or from the fact that typical Turing machines lack semantic knowledge.

I have critiqued two arguments that mental content is causally irrelevant. The literature offers many additional arguments along the same lines, some geared towards mental content in general, some geared more specifically towards wide content. I refer readers to (Rescorla 2014a) for further discussion of these matters.

12.5.2 Fodor on Implementation Mechanisms

Let us now assume that representational properties are causally relevant. Even so, intentional causation may seem rather mysterious. Perhaps there are genuinely causal generalizations that cite representational properties, but surely that is not the end of the matter. Surely we should ground intentional causal generalizations in *non-intentional* mechanisms. For example, whether a mental state represents water depends upon complex causal-historical relations to the environment. Shouldn't we isolate underlying computational mechanisms that prescind from all such causal-historical relations, citing only "local" properties of mental states?

Fodor develops this viewpoint. He holds that intentional content should figure in laws offered by scientific psychology. He also espouses a nomological theory of causal relevance, so that appropriate participation in intentional laws ensures the causal relevance of intentional properties (1990, p. 148). At the same time, he insists that intentional laws require an *implementation mechanism*: "it's got to be possible to tell the whole story about mental causation (the whole story about the implementation of the generalizations that belief/desire explanations articulate) *without referring to the intentional properties of the mental states that such generalizations subsume*" (1987, p. 139). He proposes that formal syntactic mechanisms implement intentional laws. On the resulting picture, *syntactic* mechanisms ensure the causal relevance of *intentional* properties.

To motivate his approach, Fodor (1987, p. 19) observes that the computer revolution enables us to build a machine such that

> The operations of the machine consist entirely of transformations of symbols; in the course of performing those operations, the machine is sensitive solely to syntactic properties of the symbols; and the operations that the machine performs on the symbols are entirely confined to altering their shapes. Yet the machine is so devised that it will transform one symbol into another if and only if the propositions expressed by the symbols that are so transformed stand in certain *semantic* relations --- e.g. the relation that the premises bear to the conclusion in an argument.

By positing semantically indeterminate mental syntax, we explain how mental activity respects semantic relations among mental states: "*if* the mind is a sort of computer, we begin to see how you can have a theory of mental processes… which explains how there could be nonarbitrary content relations among causally related thoughts" (1987, p. 19). CTM + FSC, unlike all rival theories, shows how mental states can participate in causal processes that "track" their meanings.

Fodor often suggests that we should individuate syntactic types by their *shapes*: "to all intents and purposes, syntax reduces to shape" (1987, p. 18), and "formal operations apply in terms of the, as it were, shapes of the objects in their domains" (1981, p. 227). Taken literally, such passages are unacceptable. As numerous commentators have observed (e.g. Bermúdez 1995, p. 364; Block 1983, pp. 521–522), internal states of an ordinary personal computer do not have shapes that are relevant to their syntax. Computation *can* be defined over syntactic types individuated partly by their shapes, but it *need not* be.

In other passages, Fodor recognizes that talk about shape is misleading (1987, p. 156, fn. 5). He envisages some more general notion of "form," not specifically tied to *geometric* form. Unfortunately, Fodor systematically equivocates between two very different positions. The first position holds that Mentalese syntactic types are individuated by *physical* or *neurophysiological* properties: "[t]okens of primitive Mentalese formulas are of different types when they differ in the (presumably physical) properties to which mental processes are sensitive" (2008, p. 79). The second position holds that Mentalese syntactic types are individuated *functionally*, so that "computational states and processes are multiply realized by neurological states (or whatever)" (2008, p. 91). On the first position, all tokens of a primitive Mentalese type share some salient physical or neurological property. On the second position, "[w]e can't take for granted that computationally homogeneous primitive Mentalese expressions *ipso facto* have neurologically homogeneous implementations; indeed, we had better take for granted that they often don't" (2008, p. 90). These two positions are inconsistent. Moreover, the first position blatantly flouts multiple realizability – one of the main virtues of CTM advertised by Fodor (1975) himself.[10]

Setting aside these equivocations, the key point here is that Fodor's *arguments* do not support multiply realizable, semantically indeterminate Mentalese types. To see why, consider an ordinary personal computer. Computational states in the computer are realized by electromagnetic states. If we program the computer appropriately, then causal interactions among electromagnetic states "track" semantic relations among corresponding computational states. For example, we can program the computer so that it carries premises only to conclusions logically entailed by those premises. To explain why computational activity respects semantic relations among computational states, we can mention correlations between electromagnetic states and semantic properties. We thereby describe the machine in *electromagnetic* terms and *representational* terms. We can also describe the machine in formal syntactic terms, but doing so is not necessary for explaining non-arbitrary content relations among the machine's causally related states. Correlations between electromagnetic states and representational states already suffice for a satisfying explanation.

A similar point applies to mental activity. Let us grant that a complete theory of mental causation must isolate a non-intentional implementation mechanism. There are at least two options:

(1) Intentional generalizations are implemented by neurophysiological processes.
(2) Intentional generalizations are implemented by formal syntactic processes.

(1) and (2) are compatible, but (2) goes beyond (1). To vindicate (1), we correlate intentional states with neural states, and we describe how transitions among neural states track transitions among intentional states. To vindicate (2), we must do more. We must introduce a formal syntactic description that applies not only to humans but also to diverse possible physically heterogeneous creatures. Perhaps we *can*

[10] For related criticisms of Fodor, see (Aydede 1999), (Prinz 2011), and (Tye and Sainsbury 2012, pp. 85–87).

isolate such a formal syntactic description. But why *should* we? Fodor has provided no sound argument that a good theory of implementation mechanisms requires (2) rather than (1).

According to Fodor, "[i]t is central to a computational psychology that the effects of semantic identities and differences on mental processes must always be mediated by 'local' properties of mental representations, hence by their nonsemantic properties assuming that semantics is externalist" (1994, p. 107). I agree that local properties of mental representations mediate the effects of semantic identities and differences. Transitions among representational mental states are not magical. Mental states are realized by neural states, and transitions among mental states are implemented by neural processes. Ultimately, the brain is just responding to local brain states. However, these uncontroversial observations do not favor (2) over (1). The "local non-semantic properties" to which a computational system responds may be neural rather than syntactic. As long as the system instantiates reliable correlations between neural and semantic properties, causal interactions among its internal states can "track" meanings in a non-arbitrary way.

Fodor wants to establish (2). But his appeal to non-intentional implementation mechanisms does not support (2) in addition to, or instead of, (1). In effect, Fodor's exposition bridges this argumentative gap by systematically equivocating between (1) and (2).

12.5.3 Chalmers on Causal Topology

Unlike Fodor, Chalmers carefully maintains the distinction between syntactic and neurophysiological description. He argues that syntactic description "yields a sweet spot of being detailed enough that a fully specified mechanism is provided, while at the same time providing the minimal level of detail needed for such a mechanism," where a "fully specified mechanism" is one that provides "a recipe that could be copied to yield a system that performs the [cognitive or behavioral] function in question" (2012, p. 245). Representational description does not fully specify a mechanism, because it does not provide an explicit recipe that one can readily convert into a physical machine. Neural explanation fully specifies a mechanism, but it includes undesirable neural details. Formal syntactic description is genuinely mechanistic (unlike representational description), and it also offers a desirable level of generality that eludes neurophysiological description.

I reply that extra generality does not necessarily promote good explanation. Suppose we want to explain why John failed the test. We might note that

> John did not study all semester.

Alternatively, we might note that

> John did not study all semester *or* John was seriously ill.

There is a clear sense in which the second explanation is more general than first. Nevertheless, it does not seem superior. This simple example illustrates a widely

recognized *problem of irrelevant disjunction*: if one boosts generality by appending irrelevant disjuncts, then no explanatory gain results (Williamson 2000). Thus, the mere fact that formal syntactic explanations offer greater generality than neuroscientific explanations does not show that they yield any explanatory advance. They may achieve greater generality only by citing surreptitiously disjunctive or gerrymandered types.

These worries exert particular force against Chalmers's approach to computational modeling. Chalmers's account hinges upon two key definitions:

- The *causal topology* of a system is "the pattern of interaction among parts of the system, abstracted away from the make-up of individual parts and from the way the causal connections are implemented" (2011, p. 337).
- A property *P* is *organizationally invariant* just in case "any change to the system that preserves the causal topology preserves *P*" (2011, p. 337).

According to Chalmers, a computational model individuates computational states by citing organizationally invariant properties. For that reason, computational explanation is much more general than neural explanation.[11]

Chalmers's notion of *causal topology* yields a widely applicable procedure for converting scientific explanations into more general explanations. Given a theory T_1 of some physical system (e.g. the digestive system), one extracts the causal topology attributed by T_1 to the system. One then constructs a new theory T_2 that describes this causal topology in organizationally invariant terms. T_1 may mention various non-organizationally-invariant properties (e.g. enzymes in the digestive system), but T_2 ignores such properties. In general, we would not regard T_2 as constituting any explanatory advance. For example, we would not hold that an organizationally invariant description of the digestive system offers any special insight into digestion. Chalmers insists that, in the special case of cognition, organizationally invariant description yields an explanatory advance (2012, p. 245). To validate this assessment, he must cite features that distinguish cognition from other phenomena (such as digestion) into which organizationally invariant description offers no special insight. Why suspect that causal topology deserves a more prominent role in the science of cognition than the science of digestion?

I raised this objection in (Rescorla 2012b, pp. 8–10). Chalmers (2012, p. 246) responds that my objection

> ignores the crucial difference between cognition and digestion: the former is an organizational invariant (setting externalism aside for now) while the latter is not. Causal topology does not suffice for digestion, so no adequate explanation of digestion wholly in terms of causal topology can be adequate. But causal topology suffices for cognition, so we can expect an explanation of cognition in terms of causal topology to be adequate. Such an explanation has the potential to cut at the joints that matter where a mechanistic explanation of cognition is concerned.

[11] Chalmers (2011) combines his analysis with a systematic theory of the *physical realization relation* between physical systems and abstract computational models. The theory leaves no room for physical realization of semantically permeated models. In (Rescorla 2013a), I criticize Chalmers on this score by citing specific examples drawn from CS. In (Rescorla 2014b), I propose an alternative theory of the physical realization relation. My alternative theory applies equally well to semantically indeterminate computational models and semantically permeated computational models.

Chalmers defends his assessment by citing Lewis's (1972) functionalist analysis of the mind, according to which "[p]sychological properties… are effectively defined by their role within an overall causal system: it is the pattern of interaction between different states that is definitive of a system's psychological properties" (Chalmers 2011, p. 339). Assuming this functionalist analysis, "[s]ystems with the same causal topology will… share their psychological properties (as long as their relation to the environment is appropriate)" (Chalmers 2011, p. 339). That, says Chalmers, is why organizationally invariant description illuminates cognition but not digestion.

Lewis proposed his functionalist approach as a piece of *conceptual analysis*. He sought to analyze ordinary psychological concepts: *belief*, *desire*, and so on. As Burge (2007, p. 376) and Putnam (1992) complain, analytic functionalism remains very poorly developed, lacking anything like the specificity one normally expects from a conceptual analysis. Advocates have yet to correlate a single mentalistic concept with a clearly defined causal or functional role. It seems unlikely that any clear, compelling example will ever emerge. Despite what Lewis claims, talk about causal or functional roles does not seem to capture the meaning of ordinary psychological discourse. Thus, Chalmers's argument rests upon an unsupported and implausible conceptual analysis of mentalistic terms.

Content externalism casts further doubt on Chalmers's argument. As Burge (2007) and Putnam (1988) emphasize, externalism raises serious difficulties for many versions of functionalism. Applied to Chalmers's theory, the core difficulty is that mere causal topology does not suffice for a physical system to instantiate desired representational properties. Suitable relations to the external physical or social environment are also required. Through various hedges and caveats, Chalmers acknowledges the threat posed by externalism. In my opinion, however, he does not fully recognize the threat's magnitude.

Consider Chalmers's claim that organizationally invariant description of a cognitive or behavioral function provides "a recipe that could be copied to yield a system that performs the function in question." Assuming that we specify the cognitive or behavioral function in externalist terms, this claim is false. For example, Burge argues that perceptual psychology routinely individuates perceptual states in externalist terms (e.g. through representational relations to specific distal shapes, sizes, or distances). Assuming that Burge is correct, a computational system that replicates the perceptual system's causal topology need not replicate relevant representational properties of perceptual states (e.g. representational relations to specific distal properties). I can concede that "causal topology suffices for cognition," and hence that an organizationally invariant duplicate of the perceptual system instantiates *some* mental activity. But this activity may differ significantly from mental activity *as described in the externalist terms employed by perceptual psychology*. Our duplicate need not perform the representational functions executed by the perceptual system (e.g. estimation of specific distal properties).

Chalmers may reply that causal topology *plus suitable embedding in the environment* suffices for desired mental activity. This reply is plausible. Also plausible is that causal topology *plus suitable embedding in the human body* suffices for diges-

tion. In neither case do we have any solid reason to believe that causal topology *taken on its own* yields a valuable level of description. The human digestive system and a causal-topological duplicate thereof need not be type-identical in any scientifically important respect. Why believe that the human mind and a causal-topological duplicate thereof are type-identical in some scientifically important respect?

Chalmers claims that we can factor out externalist elements of mental activity, leaving behind an explanatorily significant organizationally invariant remainder. He provides no reason to believe this claim except Lewis's conceptual analysis of mentalistic discourse. Once we reject Lewis's analysis, the claim no longer seems compelling. We have no reason to think that organizationally invariant description illuminates the essential natures of representational mental states, any more than it illuminates the essential natures of digestive states. For example, we have isolated no explanatorily significant respect in which states of the human perceptual system are type-identical to corresponding states of a causal-topological duplicate. Both the perceptual system and the digestive system *have* a causal topology. In neither case do we gain any evident insight by describing this causal topology, rather than *specific* causal interactions among *specific non-functional states* that instantiate the topology.

12.5.4 Formal Syntactic Mechanisms?

Proponents of CTM+RTM usually endorse two distinct theses. First, a Turing-style model of the mind decomposes mental processes into iterated elementary operations over symbols, conforming to precise, routine rules. Second, mental operations over the symbols are sensitive only to formal syntactic properties, not to semantic properties. I have disentangled the two theses by offering a view that endorses the first thesis but not the second. A Turing-style model of the mind must isolate elementary operations over mental symbols, and it must delineate rules governing how those operations are applied. But we have found no reason to assign formal syntax a privileged causal or explanatory role when describing the symbols, the operations, or the rules. For example, we have found no reason to think that rule R (as defined over semantically permeated mental symbols) requires supplementation or replacement by rule R* (as defined over semantically indeterminate syntactic types).

Quite plausibly, one must specify a non-representational implementation mechanism for a rule such as R. One must explain how the brain reliably conforms to R. But how does formal mental syntax advance this enterprise? What explanatory value does formal syntactic description contribute to a psychological theory that already contains appropriate semantically permeated computational descriptions *and* appropriate neurophysiological descriptions? Even if formal syntactic description of mental activity is possible, it may be an explanatorily idle abstraction from representational or neural description, just as organizationally invariant description of digestion would be an explanatorily idle abstraction from enzymatic description.

12.6 Turing's Legacy in the Philosophy of Mind

Widespread commitment to FSC reflects a cluster of interrelated factors: overly picturesque expositions of mathematical computation theory; distorted analyses of explanatory practice within scientific psychology; ill-motivated Quinean skepticism about intentionality; fallacious arguments that representational content is epiphenomenal; hasty appeals to implementing mechanisms for intentional laws; indiscriminate invocation of explanatory generality; underdeveloped functionalist analyses of mentalistic locutions; and so on. Once we reject these flawed arguments, we see that CTM+FSC is not well-grounded. A semantically permeated version of CTM becomes vastly more attractive than current philosophical discussion recognizes. The semantically permeated approach is not committed to supernatural entities, spooky action-at-a-distance, inner homunculi, or other mysterious posits. It simply type-identifies mental computations through their representational properties, as opposed to any alleged formal syntactic properties.

According to Fodor (2000, pp. 1–22), Turing's main contribution to cognitive science was showing how formal syntactic manipulations by a machine can respect semantic properties. I agree that this was a pivotal contribution *to computer science and Artificial Intelligence*. I think that its importance *to scientific psychology and philosophy of mind* remains undemonstrated. I would instead locate Turing's fundamental contribution to philosophy of mind elsewhere. Turing showed that iterated elementary symbolic manipulations conforming to precise, routine rules can yield astonishingly sophisticated computational activity. He thereby enabled the first mechanical models even remotely suited to accommodate paradigmatic mental processes such as reasoning, decision-making, perception, and so on. These developments conferred unprecedented substance and plausibility upon the ancient doctrine that the mind is a machine. I have shown how we can extricate this fundamental contribution from the orthodox emphasis upon formal syntactic manipulation. By doing so, we may hope to achieve a more satisfying reconciliation of two enduring pictures: *mind as machine* and *mind as representational organ*.

Acknowledgments I am indebted to audiences at UCLA, Columbia University, and the Southern Society for Philosophy and Psychology for comments when I presented earlier versions of this material. I also thank José Luis Bermúdez, Tyler Burge, Peter Carruthers, Frances Egan, Kevin Falvey, Juliet Floyd, Christopher Gauker, Mark Greenberg, Christopher Peacocke, Gualtiero Piccinini, and Daniel Weiskopf for helpful feedback.

References

Aydede, M. 1999. On the Type/Token Relation of Mental Representations. *Acta Analytica* 2: 23–50.

Aydede, M. 2005. Computationalism and Functionalism: Syntactic Theory of Mind Revisited. In: *Turkish Studies in the History and Philosophy of Science*, eds. G. Irzik and G. Güzeldere. Dordrecht: Springer.

Bays, P., and D. Wolpert. 2007. Computational Principles of Sensorimotor Control that Minimize Uncertainty and Variability. *Journal of Physiology* 578: 387–396.
Bermúdez, J.L. 1995. Syntax, Semantics, and Levels of Explanation. *Philological Quarterly* 45: 361–367.
Block, N. 1983. Mental Pictures and Cognitive Science. *Philosophical Review* 92: 499–541.
Burge, T. 2007. *Foundations of Mind*. Oxford: Oxford University Press.
———. 2010. *Origins of Objectivity*. Oxford: Oxford University Press.
Chalmers, D. 2011. A Computational Foundation for the Study of Cognition. *The Journal Cognitive Science* 12: 323–357.
———. 2012. The Varieties of Computation: A Reply. *The Journal of Cognitive Science* 13: 213–248.
Churchland, P. 1981. Eliminative Materialism and the Propositional Attitudes. *Journal of Philosophy* 78: 67–90.
Dreyfus, H. 1992. *What Computers Still Can't Do*. Cambridge: MIT Press.
Egan, F. 1992. Individualism, Computation, and Perceptual Content. *Mind* 101: 443–459.
———. 2003. Naturalistic Inquiry: Where Does Mental Representation Fit in? In *Chomsky and His Critics*, ed. L. Antony and N. Hornstein. Malden: Blackwell.
Feldman, J. 2015. Bayesian Models of Perceptual Organization. In *The Oxford Handbook of Perceptual Organization*, ed. J. Wagemans. Oxford: Oxford University Press.
Field, H. 2001. *Truth and the Absence of Fact*. Oxford: Clarendon Press.
Fodor, J. 1975. *The Language of Thought*. New York: Thomas Y. Crowell.
———. 1981. *Representations*. Cambridge: MIT Press.
———. 1987. *Psychosemantics*. Cambridge: MIT Press.
———. 1990. *A Theory of Content and Other Essays*. Cambridge: MIT Press.
———. 1991. Replies. In *Meaning in Mind*, ed. B. Loewer and G. Rey. Cambridge: Blackwell.
———. 1994. *The Elm and the Expert*. Cambridge: MIT Press.
———. 1998. *Concepts*. Oxford: Clarendon Press.
———. 2000. *The Mind Doesn't Work that Way*. Cambridge: MIT Press.
———. 2008. *LOT2*. Oxford: Clarendon Press.
Gallistel, R., and A. King. 2009. *Memory and the Computational Brain*. Malden: Wiley-Blackwell.
Harnad, S. 1994. Computation is Just Interpretable Symbol Manipulation; Cognition Isn't. *Minds and Machines* 4: 379–390.
Haugeland, J. 1985. *Artificial Intelligence: The Very Idea*. Cambridge: MIT Press.
Kaplan, D. 1989. Demonstratives. In *Themes from Kaplan*, ed. J. Almog, J. Perry, and H. Wettstein. Oxford: Oxford University Press.
Knill, D., and W. Richards, eds. 1996. *Perception as Bayesian Inference*. Cambridge: Cambridge University Press.
Lewis, D. 1972. Psychophysical and Theoretical Identifications. *Australasian Journal of Philosophy* 50: 249–258.
Ockham, W. (1957). *Summa Logicae*, in His *Philosophical Writings, A Selection*, ed. and trans. P. Boehner. London: Nelson.
Palmer, S. 1999. *Vision Science*. Cambridge: MIT Press.
Peacocke, C. 1992. *A Study of Concepts*. Cambridge: MIT Press.
———. 1994. Content, Computation, and Externalism. *Mind & Language* 9: 303–335.
Pouget, A., J. Beck, W.J. Ma, and P. Latham. 2013. Probabilistic Brains: Knowns and Unknowns. *Nature Neuroscience* 16: 1170–1178.
Prinz, J. 2011. Has Mentalese Earned Its Keep? On Jerry Fodor's *LOT 2*. *Mind* 120: 485–501.
Putnam, H. 1967. Psychophysical Predicates. In *Art, Mind, and Religion*, ed. W. Capitan and D. Merrill. Pittsburgh: University of Pittsburgh Press.
———. 1975. *Mind, Language, and Reality: Philosophical Papers*. Vol. 2. Cambridge: Cambridge University Press.
———. 1988. *Representation and Reality*. Cambridge: MIT Press.

———. 1992. Why Functionalism Failed. In *Inference, Explanation and Other Philosophical Frustrations*, ed. J. Earman. Berkeley: University of California Press.
Pylyshyn, Z. 1984. *Computation and Cognition*. Cambridge: MIT Press.
Quine, W.V. 1960. *Word and Object*. Cambridge, MA: MIT Press.
Rescorla, M. 2012a. Are Computational Transitions Sensitive to Semantics? *Australasian Journal of Philosophy* 90: 703–721.
———. 2012b. How to Integrate Representation into Computational Modeling, and Why We Should. *Journal of Cognitive Science* 13: 1–38.
———. 2013a. Against Structuralist Theories of Computational Implementation. *British Journal for the Philosophy of Science* 64: 681–707.
———. 2013b. Millikan on Honeybee Navigation and Communication. In *Millikan and Her Critics*, ed. D. Ryder, J. Kingsbury, and K. Williford. Wiley-Blackwell: Malden.
———. 2014a. The Causal Relevance of Content to Computation. *Philosophy and Phenomenological Research* 88: 173–208.
———. 2014b. A Theory of Computational Implementation. *Synthese* 191: 1277–1307.
———. 2015. Bayesian Perceptual Psychology. In *The Oxford Handbook of the Philosophy of Perception*, ed. M. Matthen. Oxford: Oxford University Press.
Salmon, N. 1986. *Frege's Puzzle*. Cambridge: MIT Press.
Schneider, S. 2011. *The Language of Thought: A New Philosophical Direction*. Cambridge: MIT Press.
Searle, J. 1980. Minds, Brains, and Programs. *Behavioral and Brain Sciences* 3: 417–424.
Smolensky, P. 1988. On the Proper Treatment of Connectionism. *Behavioral and Brain Sciences* 11: 1–74.
Stalnaker, R. 1984. *Inquiry*. Cambridge: MIT Press.
Stich, S. 1983. *From Folk Psychology to Cognitive Science*. Cambridge: MIT Press.
Taylor, R.G. 1998. *Models of Computation and Formal Languages*. Oxford: Oxford University Press.
Turing, A.. 1936/7. On Computable Numbers, with an Application to the *Entscheidungsproblem*. *Proceedings of the London Mathematical Society*, 42, 230-265.
Turing, A.M. 1947. Lecture to the London Mathematical Society on 20 February 1947. In *Mechanical Intelligence*, ed. D. Ince. Amsterdam: North-Holland.
Tye, M., and M. Sainsbury. 2012. *Seven Puzzles of Thought*. Oxford: Oxford University Press.
van Gelder, T. 1995. What Might Cognition Be, If Not Computation? *Journal of Philosophy* 7: 345–381.
Vilares, I., and K. Körding. 2011. Bayesian Models: The Structure of the World, Uncertainty, Behavior, and the Brain. *Annals of the New York Academy of Sciences* 1224: 22–39.
Williamson, T. 2000. *Knowledge and Its Limits*. Oxford: Oxford University Press.

Chapter 13
Turing and Free Will: A New Take on an Old Debate

Diane Proudfoot

Abstract In 1948 Turing claimed that the concept of intelligence is an "emotional concept". An emotional concept is a *response-dependent* concept and Turing's remarks in his 1948 and 1952 papers suggest a response-dependence approach to the concept of intelligence. On this view, whether or not an object is intelligent is determined, as Turing said, "as much by our own state of mind and training as by the properties of the object". His discussion of free will suggests a similar approach. Turing said, for example, that if a machine's program "results in its doing something interesting which we had not anticipated I should be inclined to say that the machine *had* originated something". This points to a new form of free will compatibilism, which I call *response-dependence compatibilism* and explore here.

13.1 Introduction

In recent work I have argued for a new interpretation of Turing's concept of intelligence and his test of intelligence in machines, based on his notion of an *emotional concept* and his versions of the imitation game additional to the famous version in "Computing Machinery and Intelligence" (Proudfoot 2011, 2013, 2017b). On this interpretation, Turing proposed a *response-dependence* account of the concept of intelligence; his 1948 and 1952 papers, "Intelligent Machinery" and "Can Automatic Calculating Machines be said to Think?", add intelligence to the list of putative response-dependent concepts or properties. In this chapter I apply Turing's notion of an emotional concept to his discussion of free will.

A Turing machine (for Turing, a "logical computing machine") is based upon the actual "human computer" in the process of calculating a number, of whom Turing famously said, "The behaviour of the computer at any moment is determined by the

D. Proudfoot (✉)
Department of Philosophy, University of Canterbury, Christchurch, New Zealand
e-mail: diane.proudfoot@canterbury.ac.nz

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_13

symbols which he is observing, and his 'state of mind' at that moment" (1936, p. 75). Not only the human computer's behavior, but also his or her 'state of mind' is determined; state of mind and observed symbols "determine the state of mind of the computer after the operation is carried out" (ibid., p. 77). Analogously the Turing machine's behavior is determined by the machine's "state of mind" and the scanned symbols on the square of the machine's tape: this combination "determines the possible behavior of the machine" (ibid., p. 59). This leads to the question whether a Turing machine can possess free will. In "Can Digital Computers Think?" Turing said:

> To behave like a brain seems to involve free will, but the behaviour of a digital computer, when it has been programmed, is completely determined. These two facts must somehow be reconciled, but to do so seems to involve us in an age-old controversy, that of "free will and determinism". (1951, p. 484)

Turing addressed this "controversy", directly or indirectly, in all his papers on artificial intelligence. He did not explicitly claim that human beings do have free will, or that they do not; he allowed the possibility that "the feeling of free will which we all have is an illusion" (1951, p. 484). However, on the assumption that we do have free will, the problem of reconciling the "two facts" arose for his hypothesis that "real brains, as found in animals, and in particular in men, are a sort of [Turing] machine" (ibid., p. 483). It arose also for his hypothesis that "[Turing] machines can be constructed which will simulate the behaviour of the human mind very closely" (c. 1951, p. 472).

Turing has been read as holding that the mind is indeterministic (or unpredictable, or uncomputable)—and that this is the solution to the problem of free will and determinism. (For this reading, see (Aaronson 2013), (Lloyd 2012); (Copeland 2004, 2013) offers a more cautious interpretation.) I shall argue, in contrast, that for Turing the concept of free will is an emotional concept: whether or not an agent possesses free will depends on how we respond to the agent. His discussion suggests a new form of compatibilism—a response-dependence compatibilism. This is further evidence against the standard depiction of Turing as a behaviorist.

13.2 Intelligence as an Emotional Concept[1]

In his 1948 report, "Intelligent Machinery", Turing said that "the idea of 'intelligence' is itself emotional rather than mathematical" (1948, p. 411). In a section entitled "Intelligence as an emotional concept", he wrote:

> The extent to which we regard something as behaving in an intelligent manner is determined as much by our own state of mind and training as by the properties of the object under consideration. If we are able to explain and predict its behaviour or if there seems to be little underlying plan, we have little temptation to imagine intelligence. With the same

[1] The material in this section is set out in detail in (Proudfoot 2013).

> object therefore it is possible that one man would consider it as intelligent and another would not; the second man would have found out the rules of its behaviour. (ibid., p. 431)

In places Turing used the expression "emotional" as proxy for "irrational"; he claimed that several arguments against the possibility of thinking machines are emotional in this sense (see Proudfoot 2014). An emotional *concept*, however, is not an irrational concept; it is a concept the application of which is, as Turing said here, determined "as much by our own state of mind and training as by the properties of the object under consideration". In modern terminology an emotional concept is a response-dependent concept.[2]

The notion of an emotional concept makes clear Turing's approach to intelligence. In his view, in the case of intelligence in machines, the appearance of thinking is at least as important as the machine's processing speed, storage capacity, or complexity of programming. These are examples solely of the machine's behavior—in Turing's words, the "properties of the object" rather than the properties assigned by "our own state of mind and training". In his 1952 broadcast, "Can Automatic Calculating Machines be said to Think?", Turing said "As soon as one can see the cause and effect working themselves out in the brain, one regards it as not being thinking, but a sort of unimaginative donkey-work" (1952, p. 500). Here he emphasized that intelligence is not a feature of the world independent of our tendency to "regard" entities as thinking.

This response-dependence approach is embodied in Turing's famous test of intelligence in machines. The paragraph (quoted above) beginning "The extent to which we regard something as behaving in an intelligent manner …" immediately precedes his description of the first version of the imitation game, which is restricted to chess-playing. Turing continued:

> It is possible to do a little experiment on these lines, even at the present state of knowledge. It is not difficult to devise a paper machine which will play a not very bad game of chess. Now get three men as subjects for the experiment A, B, C. A and C are to be rather poor chess players, B is the operator who works the paper machine. (In order that he should be able to work it fairly fast, it is advisable that he be both mathematician and chess player.) Two rooms are used with some arrangement for communicating moves, and a game is played between C and either A or the paper machine. C may find it quite difficult to tell which he is playing.
>
> (This is a rather idealized form of an experiment I have actually done) (1948, p. 431)

Turing's "little experiment" is a trial to see whether or not C has the "temptation to imagine intelligence" in the "paper" machine (i.e. a human executing a program).[3] Whether or not the machine is intelligent is determined in part by C's response; for example, if C can "predict its behaviour or if there seems to be little underlying plan", the machine is not judged to be intelligent. Turing's words make it clear that his game tests the *observer* rather than—as the canonical behaviorist interpretation of the imitation game assumes—the *machine.*

---

[2] On these concepts see e.g. (Johnston 1989), (Pettit 1991).

[3] A "paper machine" is a human being "provided with paper, pencil, and rubber, and subject to strict discipline", carrying out a set of rules (Turing 1948, p. 416).

Turing made it clear that he was not proposing a necessary condition of intelligence in machines (1950, p. 435), so the response-dependence theory of the concept (or property) of intelligence to be derived from his remarks can provide only a sufficient condition.[4] A naïve theory might say: *x* is *intelligent* (or *thinks*) if, in normal conditions, *x* appears intelligent to normal subjects. The central task is then to specify non-vacuous normal ("standard", "ideal", or "favorable") subjects and conditions. Turing's 1950 and 1952 versions of his imitation game implicitly indicate such subjects and conditions. The imitation-game interrogator stands in for the normal subject, and is to be "average" and "not ... expert about machines" (1950, p. 442; 1952, p. 495). The unrestricted imitation game supplies the normal conditions: a contestant is required to answer questions on "almost any one of the fields of human endeavour that we wish to include" (1950, p. 435). This requirement prevents a simplistic conversational program—for example, the program briefly claimed in mid-2014 to have passed the test, or the other contestants in Hugh Loebner's annual competition—from appearing intelligent.[5] Turing's remarks suggest, then, something like this schema: *x* is intelligent (or thinks) if, in an unrestricted computer-imitates-human game, *x* appears intelligent to an average interrogator.

In Turing's 1952 broadcast, he also emphasized—answering a lookup table objection raised by Max Newman—that his imitation game is a test of *real-world* machines (1952, p. 503). This suggests that the schema above should be modified as follows: *x* is intelligent (or thinks) if *in the actual world*, in an unrestricted computer-imitates-human game, *x* appears intelligent to an average interrogator. This is Turing's "criterion for 'thinking'" (1950, p. 436). This modification fits with the response-dependence interpretation of Turing's test, since rigid response-dependence theories world-relativize their schemas in order to eliminate objections that are based on counterfactual situations. Only the logical possibility of an unintelligent machine that can pass the test in the actual world undermines the test.

The question arises, then, whether the notion of an emotional concept is to be found elsewhere in Turing's thoughts on artificial intelligence. First, however, I turn to his explicit remarks on "free will".

13.3 Spirit and Matter

In the late 1920s and early 1930s the problem of free will and determinism was energetically and publicly debated. According to the Bishop of Birmingham, "the notion that Nature was ruled by blind mechanism was general alike in the street and

[4] A modern theory would distinguish a response-dependence theory of the property from that of the concept of intelligence, but it would be anachronistic to read Turing in this light; his remarks suggest both.

[5] For the program purportedly passing the test in mid-2014, see "Turing test success marks milestone in computing history", 8 June 2014, http://www.reading.ac.uk/news-and-events/releases/PR583836.aspx. On Turing's test and simplistic programs see (Proudfoot 2017c).

in the pew".[6] The new idea of an indeterministic universe provided a solution. The *Times* reported Sir Arthur Eddington, Plumian Professor of Astronomy at the University of Cambridge, as claiming that "[s]o far as we had yet gone in our probing of the material universe, we could not find a particle of evidence in favour of determinism. There was no longer any need to doubt our intuition of free will".[7] Eddington's theories about the universe, set out in books, lectures, and on radio, were well-known.[8] His book *The Nature of the Physical World*, based on his 1927 Gifford Lectures and published the following year, was printed five times within the first 2 years. *The Times* described it as a work "which everyone interested in the modern development of science should procure and study".[9] According to the Dean of St Paul's Cathedral, "that books on astronomy, however clearly and brilliantly written, should be reckoned among the best sellers was a very remarkable and encouraging sign. ... [T]he multitudes who read [James] Jeans and Eddington, or listened to their lectures on the wireless … rightly felt that we were in the presence of a mighty revelation of the grandeur of Nature's God".[10] Even politicians entered the debate; according to the Home Secretary, Sir Herbert Samuel, the new picture of the universe held that "at the heart of Nature pure hazard reigned", with consequences for "the freedom of the human will".[11]

In *The Nature of the Physical World* Eddington said:

> It is a consequence of the advent of the quantum that *physics is no longer pledged to a scheme of deterministic law*. … The future is a combination of the causal influences of the past together with unpredictable elements—unpredictable not merely because it is impracticable to obtain the data of prediction, but because no data connected causally with our experience exist. … [S]cience thereby withdraws its moral opposition to freewill. (1928, pp. 294–5)[12]
>
> I think we may now feel quite satisfied that the volition [i.e. "the decision between the possible behaviours" of the brain] is genuine. The materialist view was that the motions which appear to be caused by our volition are really reflex actions controlled by the material processes in the brain, the act of will being an inessential side phenomenon occurring simultaneously with the physical phenomena. But this assumes that the result of applying physical laws to the brain is fully determinate. … [T]here is nothing in the physical world

[6] As reported by *The Times*, in "Modernism As A Creed", *Times* [London, England] 11 December 1929, p. 21. *The Times Digital Archive*.

[7] As reported by *The Times*, in "Science and Religion", a report of Eddington's BBC broadcast in the "Science and Religion" series, *Times* [London, England] 24 November 1930, p. 9. *The Times Digital Archive.*

[8] See e.g. "The Expanding Universe", *Times* [London, England] 7 November 1931, p. 7. *The Times Digital Archive*.

[9] As reported in the Cambridge University Press advertisement, *Times* [London, England] 30 September 1929, p. 14. *The Times Digital Archive*.

[10] As reported by *The Times*, in "Dissolution Of The Universe", *Times* [London, England] 23 November 1931, p. 8. *The Times Digital Archive*.

[11] As reported by *The Times*, in "Philosophy For All", *Times* [London, England] 26 May 1932, p. 9. *The Times Digital Archive.*

[12] In *The Nature of the Physical World* (1948) Eddington did not distinguish an action's being predictable from its being determined (as in the just-quoted passage), nor an action's being uncaused from its being nondeterministically caused.

> … to predetermine the decision; the decision is a fact of the physical world with consequences in the future but not causally connected to the past. (ibid., p. 311)

According to Eddington, there is "no cause" of "the decision of the brain" (ibid., p. 312). Decisions are uncaused events and so are safe from the specter of determinism.[13] Volition is "something outside causality" (ibid., p. 312). Eddington conceded that his account admitted "some degree of supernaturalism" (ibid., p. 347). (We might in consequence regard his "decisions" not as *uncaused* but as examples of *substance causation*.[14])

Turing borrowed *The Nature of the Physical World* from the Sherborne School library in March and April 1929 and in May, June, and July 1930—nearly 3 months in total.[15] Andrew Hodges suggests that Turing could have found many of the ideas that he expressed in "Nature of Spirit",[16] a brief unpublished essay, in Eddington's book (Hodges 1983/2012, p. 64). However, it is not known exactly when Turing wrote "Nature of Spirit"[17] and so the most that can be said is that there are similarities between this essay's approach to free will and Eddington's view.

In "Nature of Spirit", Turing wrote:

> It used to be supposed in science that if everything was known about the universe at any particular moment then we can predict what it will be through all the future. … More modern science however has come to the conclusion that when we are dealing with atoms & electrons we are quite unable to know the exact state of them; our instruments being made of atoms & electrons themselves. The conception then of being able to know the exact state of the universe then really must break down on the small scale. This means then that the theory which held that as eclipses etc were predestined so were all our actions breaks down too.

For Turing actions are not "predestined" by preceding physical events; they are instead the result of "a will" that involves "a spirit". He said:

> Personally I think that spirit is really eternally connected with matter but certainly not always by the same kind of body. I did believe it possible for a spirit at death to go to a universe entirely separate from our own, but I now consider that matter & spirit are so connected that this would be a contradiction in terms. It is possible however but unlikely that such universes may exist.

---

[13] This claim leaves open the possibility of non-physical laws governing volition. Eddington said, "Those who maintain a deterministic theory of mental activity must do so as the outcome of their study of the mind itself and not with the idea that they are thereby making it more conformable with our experimental knowledge of the laws of inorganic nature" (1928, p. 295).

[14] On substance causation see e.g. (Lowe 2008).

[15] Turing borrowed the book on the following dates: 17–21 March 1929, 24 March-20 April 1929, and 31 May-24 July 1930. I am grateful to the School Archivist, Rachel Hassall, for this information.

[16] "Nature of Spirit" is a hand-written manuscript in the Turing Digital Archive, King's College Cambridge, catalogue reference AMT/C/29; all quotations in the text are from this manuscript. The manuscript is also transcribed in Hodges 1983/2012, pp. 63–4.

[17] Hodges suggests April 1932 (1983/2012, pp. 63, 544); the King's College Archive has this date, probably supplied by Hodges (personal communication from the KCC Archivist, Patricia McGuire).

> Then as regards the actual connection between spirit and body I consider that the body by reason of being a living body can "attract" & hold on to a "spirit", whilst the body is alive and awake the two are firmly connected & when the body is asleep I cannot guess what happens but when the body dies the "mechanism" of the body, holding the spirit is gone & the spirit finds a new body sooner or later perhaps immediately.

This is consistent with Eddington's view of volition.

What *is* "the actual connection between spirit and body"? According to Turing:

> We have a will which is able to determine the action of the atoms probably in a small portion of the brain, or possibly all over it. The rest of the body acts so as to amplify this.

This is analogous to Eddington's view. He said:

> At some brain centre the course of behaviour of certain atoms or elements of the physical world is directly determined for them by the mental decision … It seems that we must attribute to the mind power not only to decide the behaviour of atoms individually but to affect systematically large groups—in fact to tamper with the odds on atomic behaviour. (1928, pp. 312–3)

Eddington added, "This has always been one of the most dubious points in the theory of the interaction of mind and matter" (1928, p. 313).

In Turing's later writings there is no (unequivocal) reference to uncaused events. Although he remarked that the "activity of the intuition consists in making spontaneous judgments", there is no reason to think that "spontaneous" decisions are uncaused (1938, p. 192). Turing said that these judgments "are not the result of conscious trains of reasoning", which is consistent with their having causes inaccessible to consciousness (ibid., p. 192). Nor is there any serious reference in his later writings to the notion of spirit. This notion appears only in his reply to the "Theological Objection" to the possibility of thinking machines—this is the objection that thinking is "a function of man's immortal soul" (1950, p. 449). Turing's reply was that this objection "implies a serious restriction of the omnipotence of the Almighty … [who] has freedom to confer a soul on an elephant if He sees fit" (ibid., p. 449; see also Turing 1948, p. 410). This reply is rhetorical and does not imply that Turing endorsed the notion of a supernatural soul or spirit. It would seem that Turing's more considered view of free will must differ from his account in "Nature of Spirit".[18]

[18] In Hodges' view, Turing's later solution to the problem of free will and determinism is that "in practice to say anything about the world would require many different levels of description"—we describe the one phenomenon as both determined and free (1983/2012, p. 108). I cannot find any evidence for this reading of Turing. Hodges also implies that Turing shared Tolstoy's view that "what is known to us we call the laws of necessity; what is unknown we call free will. Free will is for history only an expression connoting what we do not know about the laws of human life" (ibid., p. 475). As evidence he writes, "In the January radio discussion [Turing] had said, 'Thinking is those mental processes we don't understand'" (1983/2012, p. 475). In fact, Turing said only that "one might be tempted to define thinking as consisting of 'those mental processes that we don't understand'"—and also that "I don't want to give a definition of thinking but if I had to I should probably be unable to say anything more about it than that it was a sort of buzzing that went on inside my head" (1952, pp. 500, 494). Although the former is an important remark if we are to understand Turing's conception of intelligence (see Proudfoot 2013), I can find no evidence that

13.4 A Random Element

In 1951, in his broadcast "Can Digital Computers Think?", Turing referred again to Eddington. Turing's remark about the "age-old controversy" continues as follows:

> There are two ways out. It may be that the feeling of free will which we all have is an illusion. Or it may be that we really have got free will, but yet there is no way of telling from our behaviour that this is so. In the latter case, however well a machine imitates a man's behaviour it is to be regarded as a mere sham. I do not know how we can ever decide between these alternatives but whichever is the correct one it is certain that a machine which is to imitate a brain must appear to behave as if it had free will, and it may well be asked how this is to be achieved. One possibility is to make its behaviour depend on something like a roulette wheel or a supply of radium. The behaviour of these may perhaps be predictable, but if so, we do not know how to do the prediction. (1951, p. 484)

Turing said that "it was even argued by Sir Arthur Eddington that on account of the indeterminacy principle in quantum mechanics no such prediction is even theoretically possible" (ibid., p. 483). There is no hint here of uncaused events or substance causation. If Turing's account of free will is indeterministic, it must involve *nondeterministic causation*—a "random element" rather than "spirit".

Against this reading of Turing's view of free will is the fact that in "Computing Machinery and Intelligence" Turing said:

> An interesting variant on the idea of a digital computer is a "digital computer with a random element". These have instructions involving the throwing of a die or some equivalent electronic process; one such instruction might for instance be, "Throw the die and put the resulting number into store 1000". Sometimes such a machine is described as having free will (though I would not use this phrase myself). (1950, p. 445)

The parenthesis suggests that he did *not* think that equipping a digital computer with a random element sufficed for the machine to act freely.[19] The reasons for this are not difficult to guess. Eddington had implied that positing uncaused events failed to solve the problem of free will and determinism, saying that it seemed "contrary to our feeling of the dignity of the mind" to "put it at the mercy of impulses with no causal antecedents" (1928, p. 293). Positing nondeterministic causation has a similar defect. Accounts of free will typically hold that an agent's action is free if and only if the agent *could have acted differently* or is the *ultimate origin* of the action. A machine equipped with a "roulette wheel" could have behaved differently, but only in that the spin of the wheel might have generated a different outcome—it is not that the machine could *voluntarily* have acted otherwise. Likewise the machine is the ultimate origin of its behavior, but only in that its behavior is settled by the spin of the wheel—the machine did not *freely choose* how to act. Equipping a com-

here he also had in mind free will. My response-dependence interpretation of Turing is very different from Hodges' reading (see Sects. 5–6); if the concept of free will is a response-dependent concept, it is no more a matter of description that an entity possesses free will than it is a matter of description that an object is colored. Moreover, with respect to free will in machines, what is critical is that the machine surprises us, rather than that we fail to understand "the laws of human life".

[19] Copeland also makes this point (Copeland (ed.) 2004, p. 477).

puter with a random element does not have the result that the machine's behavior is *free* in any intuitive sense.

Other remarks by Turing on including a random element in a machine concern the "education" of the machine:

> Each machine should be supplied with a tape bearing a random series of figures, e.g. 0 and 1 in equal quantities, and this series of figures should be used in the choices made by the machine. This would result in the behaviour of the machine not being by any means completely determined by the experiences to which it was subjected, and would have some valuable uses when one was experimenting with it. By faking the choices made one would be able to control the development of the machine to some extent. One might, for instance, insist on the choice made being a particular one at, say, 10 particular places, and this would mean that about one machine in 1024 or more would develop to as high a degree as the one which had been faked. (c. 1951, p. 475)

He also advocated using a random element as a shortcut in calculation:

> A random element is rather useful when we are searching for a solution of some problem. Suppose for instance we wanted to find a number between 50 and 200 which was equal to the square of the sum of its digits, we might start at 51 then try 52 and go on until we got a number that worked. Alternatively we might choose numbers at random until we got a good one. … Since there is probably a very large number of satisfactory solutions the random method seems to be better than the systematic. (1950, p. 463)

Turing also thought that the random method is used in "the analogous process of evolution" (ibid., p. 463).[20] There is no evidence in any of these remarks that he thought that a partially random machine possesses free will.

The principal reason to deny that Turing suggested indeterminism as the route to free will is that, after saying "One possibility is to make [the machine's] behaviour depend on something like a roulette wheel or a supply of radium", he remarked:

> It is, however, not really even necessary to do this. It is not difficult to design machines whose behaviour appears quite random to anyone who does not know the details of their construction. (1951, p. 485)

This is Turing's "apparently partially random" machine—a machine that

> may be strictly speaking determined but appear superficially as if it were partially random. This would occur if for instance the digits of the number π were used to determine the choices of a partially random machine, where previously a dice thrower or electronic equivalent had been used. (1948, p. 416)

If an *apparently* partially random machine is Turing's response to the "age-old controversy" of free will and determinism, he must have embraced some solution other than indeterminism.

[20] Turing hypothesized that the behavior of the infant brain is partially random (1948, p. 424). For a discussion of randomness and uncomputability in connection with Turing's views on the mind, see (Copeland and Shagrir 2013).

13.5 Free Will as an Emotional Concept

In describing and addressing this "controversy" the specific language of "free will" is not essential. The issue is whether an action's having the property that makes it free—for example, being produced by an agent able to act otherwise—is compatible with the action's being determined. The critics (of the possibility of thinking machines) that Turing challenged employed a variety of expressions to discuss exactly this issue.

Geoffrey Jefferson, a participant in Turing's 1952 radio discussion and cited in "Computing Machinery and Intelligence", said:

> It can be urged, and it is cogent argument against the machine, that it can answer only problems given to it, and, furthermore, that the method it employs is one prearranged by its operator. … It is not enough … to build a machine that could use words (if that were possible), it would have to be able to create concepts and to *find for itself* suitable words in which to express additions to knowledge that it brought about. (1949, pp. 1109–10)

Jefferson's criticism that the machine's method is "prearranged by its operator" is the complaint that the machine could not have acted otherwise. Likewise, his requirement that the machine "create concepts" and "find for itself" words is the demand that the machine be the true source of its behavior. That free will is Jefferson's concern is demonstrated by his saying, regarding the nervous system as opposed to "modern automata", that "although much can be properly explained by conditioned reflexes and determinism (in which idea mechanism lurks in the background), there is a fringe left over in which free will may act (i.e. choice not rigidly bound to individual precedent)" (1949, p. 1107).[21]

Likewise Ada Lovelace, also cited in "Computing Machinery and Intelligence", claimed that the Analytical Engine could not "originate" anything. She wrote:

> The Analytical Engine has no pretensions whatever to *originate* anything. It can do whatever *we know how to order it* to perform. It can *follow* analysis; but it has no power of *anticipating* any analytical relations or truths. Its province is to assist us in making *available* what we are already acquainted with. (Lovelace 1843, p. 722)

Turing quoted the first sentence of this remark, as quoted by Douglas Hartree in his *Calculating Instruments and Machines*. Hartree said, of "equipment such as electronic calculating machines, automatic pilots for aircraft", that such a machine cannot "think for itself": "all the thinking has to be done beforehand by the designer and by the operator who provides the operating instructions for the particular problem; all the machine can do is to follow these instructions exactly" (1949, p. 70). Lovelace and Hartree, like Jefferson, discussed the problem of free will and determinism without using the expression "free will".

Turing's approach can be gleaned, then, not only from his explicit remarks on "free will" but also from his responses to these criticisms. He focused on the

[21] Jefferson hinted at nondeterministic causation as providing the "fringe left over in which free will may act", saying "I accept here the emendation of Niels Bohr, who sees this as the counterpart of the impossibility of fully describing the electron either as a point or as a wave" (1949, p. 1107).

question whether a machine could be the ultimate origin of its behavior. His response to Lady Lovelace's objection was to say that this "whole question will be considered again under the heading of learning machines" (1950, p. 455; on Turing's "child machines" see Proudfoot 2017a).

On the question whether a learning machine could be intelligent, Turing said:

> One can imagine that after the machine had been operating for some time, the instructions would have altered out of all recognition, but nevertheless still be such that one would have to admit that the machine was still doing very worthwhile calculations. … In such a case one would have to admit that the progress of the machine had not been foreseen when its original instructions were put in. It would be like a pupil who had learnt much from his master, but had added much more by his own work. When this happens I feel that one is obliged to regard the machine as showing intelligence. (1947, p. 393)

In Turing's view, a machine can be intelligent (or think) regardless of whether its behavior is determined. The same applies to the question whether a machine is the ultimate origin of its behavior. He said:

> Certainly the machine can only do what we *do* order it to perform, anything else would be a mechanical fault. But there is no need to suppose that, when we give it its orders we know what we are doing, what the consequences of these orders are going to be. … If we give the machine a programme which results in its doing something interesting which we had not anticipated I should be inclined to say that the machine *had* originated something, rather than to claim that its behaviour was implicit in the programme, and therefore that the originality lies entirely with us. (1951, p. 485)

The machine can be the true source of its behaviour even if its behaviour is determined.

Using Turing's approach, whether or not an entity possesses free will is determined (at least in part) by how the observer *responds*. He said, "We should be pleased when the machine surprises us, in rather the same way as one is pleased when a pupil does something which he had not been explicitly taught to do" (1951, p. 485). The concept of free will, like that of intelligence, is an emotional concept: whether or not an entity possesses free will is determined "as much by our own state of mind and training" as by the entity's response-independent properties.

A response-dependence approach to the concept of intelligence provides a philosophical justification for Turing's use of the imitation game—that is, for his move from the question *Can machines think?* to the question *Are there imaginable digital computers which would do well in the imitation game?* Turing made an analogous move in discussing free will. A variant of Lady Lovelace's objection, he claimed,

> says that a machine can never "take us by surprise". This statement … can be met directly. Machines take me by surprise with great frequency. This is largely because I do not do sufficient calculation to decide what to expect them to do, or rather because, although I do a calculation, I do it in a hurried, slipshod fashion, taking risks. (1950, pp. 455–6)

Turing in effect replaced the question *Can a machine possess free will?* with the question *Can a machine take us by surprise?* A response-dependence approach to the concept of free will provides a philosophical justification for this move. It also explains why for Turing "it is certain that a machine which is to imitate a brain must appear to behave as if it had free will"—and why an *apparently* random machine

will do in place of a genuinely random machine (with respect to free will). *Appearing* free is what matters to *being* free.

Copeland offers a different explanation of the apparently partially random machine's role in Turing's discussion of free will. He says:

> Apparently partially random machines imitate partially random machines. As is well known, Turing advocated imitation as the basis of a test—the Turing test—that "[y]ou might call … a test to see whether the machine thinks". An appropriately programmed digital computer could give a convincing imitation of the behaviour produced by a human brain even if the brain is a partially random machine. The appearance that this deterministic machine gives of possessing free will is, Turing said, "mere sham", but it is in his view nevertheless "not altogether unreasonable" to describe a machine that successfully "imitate[s] a brain" as itself being a brain.
>
> Turing's strategy for dealing with what can be termed the *freewill objection* to human-level AI is elegant and provocative. ((2013), p. 657; see also (Copeland 2000), pp. 30–31)

However, Turing was inclined to say that a machine (that does something interesting and unexpected) "*had* originated something". This suggests that in his view a machine *can* be the ultimate origin of its behavior—rather than, as Copeland suggests here, merely give a "convincing imitation" of an entity with free will. Copeland's interpretation does not take into account Turing's notion of an emotional concept. If the concept of free will is a response-dependent concept, a machine's appearing to possess free will is *not* "mere sham" just because the machine is deterministic—any more than an object's looking red is mere sham just because its particles lack color. Likewise, it is *not* merely "not altogether unreasonable" to describe this machine as free—any more than it is merely not altogether unreasonable to describe an object that looks red (by normal subjects in normal conditions) as red.

Turing did not expect his way of dealing with Lady Lovelace's objection—substituting *Can a machine take us by surprise?* for *Can a machine possess free will?*—"to silence my critic". The critic, he said,

> will probably say that such surprises are due to some creative mental act on my part, and reflect no credit on the machine. This leads us back to the argument from consciousness, and far from the idea of surprise. It is a line of argument we must consider closed (1950, p. 456)

The "argument from consciousness" is the claim that a machine's mere behavior is not indicative of consciousness or thought. Turing "closed" this line of argument, he believed, by issuing a challenge. He argued that the options are: either behavior *is* a sign of consciousness or "the only way to know that a *man* thinks is to be that particular man" (ibid., p. 452). The latter, he said, is "the solipsist point of view" and he assumed that most people would accept his imitation game as a test of thinking rather than be committed to solipsism (ibid., p. 452). Perhaps Turing thought that the objection to the possibility of machines with free will is simply the argument from consciousness in another guise. Alternatively, perhaps he thought these objections analogous and susceptible to the same reply. Turing said that *if* "there is no way of telling from our behaviour" that "we really have got free will" then a machine's behavior "is to be regarded as a mere sham" (Section 4). Mimicking his

rejoinder to the argument from consciousness, we might say: either (surprising and interesting) behavior *is* a sign of free will or the only way to know that a *human being* possesses free will is to be that human being. With respect to thinking, Turing predicted that we would reject solipsism and accept the alternative—in this case, that is to admit that a machine's appearing to possess free will is *not* "mere sham".

13.6 Response-Dependence Compatibilism

A response-dependence approach to the concept of free will would be formulated along these lines: *x* has *free will* (or acts *voluntarily*) if and only if, in actual normal conditions, *x* appears to have free will to actual normal subjects.

Turing's remarks point to both normal subjects and conditions in the case of machines. The child machine's "teacher" is proxy for the normal subject—an experimenter who is *ignorant of the machine's engineering*, just as a teacher is ignorant of the structure and function of the human child's brain. Turing suggested that "the education of the machine should be entrusted to some highly competent schoolmaster who is interested in the project but who is forbidden any detailed knowledge of the inner workings of the machine" (c. 1951, p. 473). He said:

> An important feature of a learning machine is that its teacher will often be very largely ignorant of quite what is going on inside, although he may still be able to some extent to predict his pupil's behaviour. … The view that "the machine can only do what we know how to order it to do" appears strange in face of this. (1950, p. 462)

This parallels Turing's requirement, with respect to the concept of intelligence, that the imitation-game interrogator be "average" and "not ... expert about machines".

The teaching situation also provides normal conditions. The experimenter is surprised by the machine's behavior only against a *background of training* the machine, just as in the case of teaching the human infant. According to Turing, the experimenter is to start from "a comparatively simple machine, and, by subjecting it to a suitable range of "experience" transform it into one which was more elaborate, and was able to deal with a far greater range of contingencies. … This might be called 'education'" (c. 1951, p. 473).[22] The aim is to produce "initiative":

> [D]iscipline is certainly not enough in itself to produce intelligence. That which is required in addition we call initiative. … Bit by bit one would be able to allow the machine to make more and more "choices" or "decisions". [Eventually] interference would no longer be necessary, and the machine would have "grown up". (1928, pp. 429–30)

[22] Turing added: "This process could probably be hastened by a suitable selection of the experiences to which [the machine] was subjected. This might be called 'education'. But here we have to be careful. It would be quite easy to arrange the experiences in such a way that they automatically caused the structure of the machine to build up into a previously intended form, and this would obviously be a gross form of cheating, almost on a par with having a man inside the machine." (c. 1951, p. 473).

The machine that has "grown up", makes "choices" and "decisions", and acquires "initiative" is the ultimate source of its own behavior. Whereas in "Nature of Spirit" free will seemingly involves the supernatural, in Turing's later writings whether an agent's action is free depends on how we *react* to the agent.

If the concept of intelligence is response-dependent, it is very different from the concept of computation, *even if* brain processes implementing computations are the physical basis of "thinking" behavior. In analogous fashion, if the concept of free will is a response-dependent concept, it is very different from the concept of indeterminism—involving either an uncaused "will" or a "random element"—*even if* there is an indeterminism in the (physical or non-physical) basis of "free" behavior.[23] This point applies more generally. Libertarian and compatibilist accounts of free will argue that some response-independent property of an agent (or the agent's behavior) suffices for free will: nondeterministic causation, the role of internal states or second-order reflective mental states, the absence of impediments to action, responsiveness to reasons, and so on. If the concept of free will is an emotional concept, no observer-independent feature suffices for free will—regardless of whether the feature is internal or external to the agent.

Is a response-dependence approach to the concept of free will a form of *illusionism*?[24] Response-dependence theorists typically distinguish their accounts from subjectivist or illusionist accounts; for example, Mark Johnston claims that response-dependence is consistent with a "qualified" realism (1989, p. 148) and Philip Pettit that "response-dependence does not compromise realism in a serious manner" (1991, p. 588). The notion of normal subjects and conditions is intended precisely to provide an objective basis for the application of a response-dependent concept; also, judgments of response-dependent concepts are held by many proponents of response-dependence accounts to be genuinely evaluable as true or false (Johnston 1989, p. 148). If these theorists' arguments are sound, both intelligence and free will can be "real"—even if the concepts of intelligence and free will are response-dependent.

Turing's discussion of the emotional concept of intelligence is consistent with this approach to response-dependent concepts (see Proudfoot 2013). An emotional concept is certainly not, in his view, merely subjective, since he said that the application of an emotional concept is determined in part by its response-*in*dependent properties ("the properties of the object under consideration"). He provided normal subjects and conditions to exclude aberrant judgments of intelligence in machines. Moreover, he referred to the question *Are there imaginable digital computers which would do well in the imitation game?* as a "variant" of the question *Can machines think?* (1950, p. 442). The former is evaluable as genuinely true or false, and so (we can presume) is the latter. There is no reason to think that Turing took a different stance with respect to the emotional concept of free will.

[23] Even if the concept of free will is response-dependent, it may be that in fact only entities with certain observer-independent properties—for example, being equipped with a "random element"—generate the appropriate response in observers.

[24] On free will illusionism see e.g. (Smilansky 2001).

If the concept of free will is a response-dependent concept, the "age-old controversy" of free will and determinism does not undermine Turing's groundbreaking hypotheses about the mind. Instead the tension between determinism and free will is simply another example of the perplexing relation between primary and secondary qualities: just as an object can be colored even if its particles are not, so an action can be free even if it is also determined. The relation is certainly still to be explained, but does not (without considerable argument) rule out the possibility of color—or of free will. This leaves, as the likely other main concern about a response-dependence approach to free will, the close link between free will and moral responsibility. Does morality demand a more substantial account of free will? However, this is hardly a new concern, and proponents of anti-metaphysical accounts of free will or anti-realist ethical theories claim that such approaches are consistent with full-fledged accountability.

13.7 Conclusion

Can machinespossess free will? Turing said, "It might be argued that there is a fundamental contradiction in the idea of a machine with intelligence. It is certainly true that 'acting like a machine' has become synonymous with lack of adaptability" (1947, p. 393). The same might be said of the idea of a machine with free will. If, however, the concepts of intelligence and free will are response-dependent concepts along the lines suggested by Turing's remarks, there is no "fundamental contradiction" in the idea of a machine with intelligence or free will. Whether computers will pass Turing's test of intelligence or will "take us by surprise" is an open question.

Turing's own view is cautiously expressed. On intelligence, he said "Of course I am not saying at present either that machines really could pass the test, or that they couldn't" (1952, p. 495). Yet he also made it clear that in his view some machine *will* do well in the imitation game (e.g. 1950, p. 449). His discussion of free will—in humans and machines—follows the same pattern. As we have seen, he said "It may be that the feeling of free will which we all have is an illusion". Yet he also said that a "grown up" child or machine can possess "initiative". He rejected "the view that the credit for the discoveries of a pupil should be given to his teacher", saying:

> In such a case the teacher would be pleased with the success of his methods of education, but would not claim the results themselves unless he had actually communicated them to his pupil. (1948, pp. 411–12)

For Turing, both humans and machines *can* be the ultimate source of their behavior, even if this behavior is also determined.

Acknowledgments I am indebted to Jack Copeland and to Juliet Floyd for valuable commentary on an earlier draft of this chapter.

References

Aaronson, S. 2013. *The Ghost in the Quantum Turing Machine*. arXiv:1306.0159v2 [quant-ph] 11 Oct 2013.

Copeland, B.J. 2000. Narrow versus Wide Mechanism. *Journal of Philosophy* 96 (1): 5–32.

———., ed. 2004. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life, Plus the Secrets of Enigma*. Oxford: Oxford University Press.

———. 2013. Turing and the Physics of the Mind. In *Alan Turing: His Work and Impact*, ed. S.B. Cooper and J. van Leeuwen, 651–666. Amsterdam: Elsevier.

Copeland, B.J., and O. Shagrir. 2013. Turing versus Gödel on Computability and the Mind. In *Computability: Turing, Gödel, Church, and Beyond*, ed. B.J. Copeland, C.J. Posy, and O. Shagrir, 1–33. Cambridge, MA: MIT Press.

Eddington, A.S. 1928. *The Nature of the Physical World*. Cambridge: Cambridge University Press. Reprinted in 1948.

Hartree, D.R. 1949. *Calculating Instruments and Machines*. Urbana: University of Illinois Press.

Hodges, A. 1983/2012. *Alan Turing: The Enigma* (The Centenary Edition). London: Vintage.

Jefferson, G. 1949. The Mind of Mechanical Man. *British Medical Journal* 1 (4616): 1105–1110.

Johnston, M. 1989. Dispositional Theories of Value. *Proceedings of the Aristotelian Society, Suppl. Vol.* 63: 139–174.

Lloyd, S. 2012. A Turing Test for Free Will. *Philosophical Transactions of the Royal Society A* 370: 3597–3610.

Lowe, E.J. 2008. *Personal Agency: The Metaphysics of Mind and Action*. Oxford: Oxford University Press.

Lovelace, A.A. 1843. Notes by the Translator (addenda to her translation of L.F. Menabrea, "Sketch of *The Analytical Engine* invented by Charles Babbage"). In *Scientific Memoirs, Selected from the Transactions of Foreign Academies of Science and Learned Societies, and from Foreign Journals, Vol. 3*, ed. R. Taylor, 691–731. London: Richard and John E. Taylor.

Pettit, P. 1991. Realism and Response-dependence. *Mind* 100 (4): 587–626.

Proudfoot, D. 2011. Anthropomorphism and AI: Turing's Much Misunderstood Imitation Game. *Artificial Intelligence* 175 (5–6): 950–957.

———. 2013. Rethinking Turing's Test. *Journal of Philosophy* 110 (7): 391–411.

———. 2014. Turing's Three Senses of "Emotional". *International Journal of Synthetic Emotions* 5 (2): 7–20.

———. 2017a. Turing's Child Machines. In *The Turing Guide*, Jack Copeland et al., 315–325. Oxford: Oxford University Press.

———. 2017b. Turing's Concept of Intelligence. In *The Turing Guide*, Jack Copeland et al., 301–307. Oxford: Oxford University Press.

———. 2017c. The Turing Test–from Every Angle. In *The Turing Guide*, Jack Copeland et al., 287–300. Oxford: Oxford University Press.

Smilansky, S. 2001. Free Will: From Nature to Illusion. *Proceedings of the Aristotelian Society* 101: 71–95.

Turing, A.M. 1936. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2* (42): 230–265. Reprinted in Copeland (ed.) (2004), 58–90; page references are to Copeland (ed.) 2004.

———. 1938. Systems of Logic Based on Ordinals, Published in 1939 in the *Proceedings of the London Mathematical Society, Series* 2 (45): 161–228. Reprinted in Copeland (ed.) (2004), 146–204; page references are to Copeland (ed.) 2004.

———. 1947. Lecture on the Automatic Computing Engine. Lecture to the London Mathematical Society, 20 February 1947, in Copeland (ed.) (2004), 378–94; page references are to Copeland (ed.) 2004.

———. 1948. Intelligent Machinery. A Report Written for the National Physical Laboratory. Reprinted in Copeland (ed.) (2004), 410–432; page references are to Copeland (ed.) (2004).

———. 1950. Computing Machinery and Intelligence, *Mind* 59, 433–460. Reprinted in Copeland (ed.) (2004), 441–64; page references are to Copeland (ed.) (2004).

———. 1951. Can Digital Computers Think? Radio broadcast on BBC Radio 15 May, 1951, in Copeland (ed.) (2004), 482–486.

———. c. 1951. Intelligent Machinery: A Heretical Theory. In Copeland (ed.) (2004), 472–475.

Turing, A.M., Braithwaite, R., Jefferson, G., and Newman, M. 1952. Can Automatic Calculating Machines Be Said to Think? Radio broadcast 14 January 1952, in (Copeland (ed.) 2004), 494–506.

Chapter 14
Turing on the Integration of Human and Machine Intelligence

Susan G. Sterrett

Abstract Philosophical discussion of Alan Turing's writings on intelligence has mostly revolved around a single point made in a paper published in the journal *Mind* in 1950. This is unfortunate, for Turing's reflections on machine (artificial) intelligence, human intelligence, and the relation between them were more extensive and sophisticated. They are seen to be extremely well-considered and sound in retrospect. Recently, IBM developed a question-answering computer (Watson) that could compete against humans on the game show *Jeopardy!* There are hopes it can be adapted to other contexts besides that game show, in the role of a collaborator of, rather than a competitor to, humans. Another, different, research project—an artificial intelligence program put into operation in 2010—is the machine learning program NELL (Never Ending Language Learning), which continuously 'learns' by 'reading' massive amounts of material on millions of web pages. Both of these recent endeavors in artificial intelligence rely to some extent on the integration of human guidance and feedback at various points in the machine's learning process. In this paper, I examine Turing's remarks on the development of intelligence used in various kinds of search, in light of the experience gained to date on these projects.

14.1 Introduction: Isolation, Interference, and Immersion

In his 1948 technical report "Intelligent Machinery: A Report by A. M. Turing", in the course of exploring an analogy between the *education of a human*, where the education is designed to allow a human to achieve his or her intellectual potential, and the *education of a machine*, where the education is designed to result in a machine analogue of it, i.e., in allowing a machine to achieve *its* intellectual potential, Turing makes the observation that "an isolated [human] does not develop any intellectual power."[1] (Turing 2004a, pp. 439–440; Sterrett 2012) What he means

[1] Where I think Turing was indicating "human" or "humans" in using the term "man" or "men", I may paraphrase or replace words within brackets accordingly.

S.G. Sterrett (✉)
Department of Philosophy, Wichita State University, Wichita, KS 6720, USA
e-mail: susan.sterrett@wichita.edu

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_14

here by "an isolated [human]" is a human who is isolated from other humans, not a human isolated from contact with everything else in the world.

Turing's remark about the intellectual possibilities of humans isolated from other humans is especially notable, since it is made in a paper that also discusses trial and error approaches to learning. Given the parallels often drawn between trial and error methods of learning and adaptation by natural selection, one might expect that, instead of this comment that a human in isolation from other humans does not develop any intellectual power, an investigation into the potential that trial and error approaches hold. That is, one might expect a more sanguine investigation into what trial and error methods are able to effect (i.e., bring about) for an individual exploring the world on its own, so to speak. Such research projects are concerned with asking: what kind of priming with what kind of examples and what kind of input devices does a machine need [in order to be able to perform a certain task autonomously]? There are approaches to machine learning on which learning of certain kinds can take place due to the interaction of one individual system or machine and its environment, even if the immediate environment with which it interacts happens to be devoid of humans and of others like itself. Some unsupervised machine learning algorithms, such as clustering algorithms, are examples of this.[2] But, upon close examination of Turing's examples, it is clear that the kind of learning under consideration in his investigations is not of this sort.

Rather, in Turing's writings on intelligent machinery, the methods he considers for providing the machine analogue of the education of a human all seem to involve human interaction with the machine in some way or another, and for a significant part of the machine's development process. Sometimes the human is involved in a teaching role, i.e., actively and intentionally interacting with the child-machine in order to effect behavior of a certain sort: Turing mentions both a method using reinforcement via inputs for analogues of "pain" and "pleasure" (Sections 10 and 11 of the paper (Turing 2004a, p. 433 ff.)) and a method using the programming of "principles." (Section 12 (Turing 2004a, p. 438)). However, such explicit training of the machine by humans is not the only way of integrating human expertise via the 'education' of a machine. In discussing various stages of the education of a human (and, analogously, how he imagines the education of a machine might proceed), Turing considers the phenomenon of humans learning irregular verbs: "By long experience we can pick up and apply the most complicated rules without being able to enunciate them at all." He suspects some kinds of machines he considers (e.g., the P-type machine, an "unorganized" machine he proposed to train using machine analogues of pleasure and pain inputs delivered by a human), of behaving similarly, due to their "randomly distributed memory units." The key remark he makes here is that "built-in teaching procedures could not help." (Turing 2004a, p. 438) Yet his hope is

[2] In *supervised learning*, all of the examples given the computer program during training are labeled; in *semi-supervised learning*, the computer program is provided with labels for only a subset of the examples it is given to deal with, and in *unsupervised learning*, none of the examples the computer program is to deal with is labeled.

not empty; it amounts to the hope of there being a machine analogue of the human ability to "pick up and apply" rules—even "the most complicated" rules.

14.2 Teaching Searching

What is the human process for which Turing hopes there is a machine analogue, if not "built-in teaching procedures"? That might depend on the type of intelligent machine behavior under discussion. Turing argues for the generality of search problems, i.e., that a "very great variety of problems" displaying initiative can be characterized as problems whose solutions are solutions to a search problem. To get a general idea of his thinking here; he gives an example and argues:

> . . . the problem is clearly equivalent to that of finding a program to put on the machine in question, and it is easy to put the programs into correspondence with the positive integers in such a way that given either the number or the program the other can easily be found. We should not go far wrong for the time being if we assumed that all problems [requiring some sort of initiative] are reducible to this form. (Turing 2004a, pp. 438–439)

I take it that by "this form" of a problem, he means the form of a problem such that searching for the right program is a matter of finding, or computing, the positive integer that corresponds to it. The details of his argument can be put aside for our purposes; what is of interest for us here is that he goes on to distinguish three types of problem, each of which can be characterized as a variety of search. The three varieties of search he identifies in his discussion, listed below in the order in which he mentions them in that discussion, are:

(a) *intellectual search*,
(b) *genetical or evolutionary search*, and
(c) *cultural search.*

Intellectual search appears to be a matter of insightfully reformulating or "transforming" problems so that mathematical and logical methods can be used to organize and carry out the search for a solution to the problem at hand. Turing gave a brief definition of intellectual searches: "They might very briefly be defined as 'searches carried out by brains for combinations with particular properties'." The kind of problem just referred to in the preceding quote (a problem of finding a program with certain features) is an example of intellectual search.

Now, as to how humans are involved in the development of a machine capable of intellectual searches: It appears that what Turing thinks is that, for a single machine, one good human teacher is all that is needed. Turing speaks of how one would work with a universal machine to get it to the point where it could carry out intellectual searches; " [W]e first put a program into it which corresponds to building in a logical system (like Russell's *Principia Mathematica*). This would not determine the behaviour of the machine completely: at various stages more than one choice as to the next step would be possible." (Turing 2004a, p. 439)

He goes on to explain how a human could interact with such a machine in a way that might be expected to eventually result in a machine capable of carrying out intellectual searches; the details are not important to us here. What is important is this: even if the machine has no other interactions with humans than this one human teacher, there is no question of the entire process occurring completely outside of, or in isolation of, human society: after all, the teacher is not isolated from human society. Turing recognizes the crucial importance of human society in "organizing", or educating a human; he seems to think that it is *in the interaction of a human with other humans*, rather than in the physiology of the brain or the genetics that gives rise to it, that the organization of the cortex responsible for human intelligence lies. It is in the training of the cortex that what is responsible for human intelligence arises. That one should not look to physiology alone for the key to human intelligence is brought out in a sort of thought experiment:

> … the possession of a human cortex (say) would be virtually useless if no attempt was made to organize it. Thus if a wolf by a mutation acquired a human cortex there is little reason to believe that he would have any selective advantage. If however the mutation occurred in a milieu where speech had developed (parrot-like wolves), and if the mutation by chance had well permeated a small community, then some selective advantage might be felt. It would then be possible to pass information on from generation to generation. However this is all rather speculative. (Turing 2004a, p. 433)

What Turing has done here with the oft-repeated claim that the difference between humans and animals is a matter of being able to speak bears emphasis. In these understated remarks, he imagines what it would take for animals to have speech in a sense that would have significant consequences. It is noteworthy that he distinguishes between a wolf with the physiological ability to speak and a small community of wolves that uses speech to pass information on from generation to generation; only in the latter case would the capability for speech make enough difference for it to be preserved by means of natural selection. Turing does not venture a guess here as to what kind of intellectual abilities would arise in such a community of parrot-like wolves in which speech had conferred some selective advantage but, as wolves live in packs and hunt as a team, it is not hard to imagine the kinds of behaviors in a pack of parrot-like wolves for which speech might confer some selective advantage.

14.3 Natural Searching

The issue of selective advantage does arise in the next variety of search he discusses, (b) *genetical or evolutionary search*, which is briefly described as: "a combination of genes is looked for, the criterion being survival value." Here, unlike some other commentators on this passage (Koza et al. 1999), I take Turing to be speaking about genetical or evolutionary search per se, i.e., literally, rather than conflating the point he makes in this passage with other points in later papers (e.g., his 1950 "Computing Machinery and Intelligence") in which he is explicit that he is only invoking *an*

analogy with natural selection. I take his remark here that "The remarkable success of this search confirms to some extent the idea that intellectual activity consists mainly of various kinds of search." to mean that, since the approach of regarding something as incredibly involved as finding a biological organism with certain features as a kind of search in a combinatorial space had turned out to have been successful in recent scientific research, that his idea that intellectual activity consists mainly in search had gained in plausibility as a result. Thus, "genetic or evolutionary search" may be a sort of model or ideal of the kind of search he means, but it is clearly distinguished from it: when characterized as a search, it is a search for an organism having a combination of features that results in the organism having high "survival value." In the 1950 paper, what we find is an analog of it: there, he explicitly identifies the analog of natural selection as "Judgment of the experimenter" (Turing 2004b, p. 469) and the surrounding text indicates that the experimenter will keep trying different machines to see how well each one learns. As this experimenter would be selecting machines for their value in terms of learning ability, it is not the same kind of search in terms of what is being selected for as we find in this 1948 report, "Intelligent Machinery."

Besides the reference to selective advantage in the pack of parrot-like wolves in the 1948 paper, and the other references to genetics ("It clearly would not require any very complex system of genes to produce something like the A- or B- type unorganized machine" (Turing 2004a, p. 433) in the 1948 paper, I think there are other good reasons to take Turing literally here, and these are based on looking at the historical context of the work. The historical context with respect to genetics is that, in the summer of 1948, when Turing's "Intelligent Machinery" paper was written,[3] the scientific research community in genetics was attempting a return to normalcy following the war. In 1948, the eighth meeting of the International Congress on Genetics was held, and it was a huge affair, both in size and in significance.

It was the first time the Congress had convened since 1939, the delay being due not only to the interruption of the war, but also to political issues in the science of genetics. It was the first such meeting that had been held since the "evolutionary synthesis" of the science of evolution by natural selection and the science of genetics had been fulfilled, and the potential that being able to examine and manipulate genetic material might hold in light of the synthesis was just being sketched out. At that 1948 conference, Muller, the president, spoke (among many other things) about the ability to extract and study in vitro genetic material from bacteria. Concerning genes themselves, Muller suggested that all genes probably "have the same essential composition, inasmuch as they require a combination of some kind of nucleic

[3] The dating of the work as composed in the summer of 1948 is per Jack B. Copeland, footnote 53 of Copeland (ed.). (2004), p. 409. Copeland notes errors by others in stating the date of this report. He notes that the phrase "Manchester machine (as actually working 8/7/48)" appears in both the finished NPL [National Physical Laboratory] report and in the draft typescript. Thus the draft report was completed sometime after July 8. The 1948 conference of the International Congress of Genetics was held July 7–14, 1948 (Bengtsson and Tunlid 2010, p. 709).

acid or nucleic acid prototype with some kind of protein or protein prototype".[4] Muller's speculation about the composition of genes themselves was framed in terms of combinations: combination, that was the key in 1948.

Once Turing's comment about "genetical or evolutionary" search here is placed in its historical context, there does not seem to be any reason not to take at face value this comment by one of the most outstanding cryptographers of the time, to the effect that he regards the task of finding the genetic code associated with features of an organism that increase its "survival value" to be a type of search [among different possible combinations]. I think his point in identifying genetical or evolutionary search as a distinct kind of search is that, in light of how well the evolutionary synthesis had turned out by 1948 (from the standpoint of those judging it in 1948), genetical or evolutionary science seemed finally to conclusively support the idea that natural selection, too, can be seen as a kind of search, a search for the right combination of genes. It seemed to portend great advances, too. It seems to me to be something of a precursor of a way of viewing natural selection now known as the "selfish gene" view. Writing over 65 years later, such a "selfish gene" view is now no longer novel, nor is it now unquestioned. The current status of this view in biological science does not matter to our discussion in this paper, though, which is understanding Turing's points about machine intelligence.

The point about genetic search relevant to us here—and to the point of the essay on intelligent machinery in which it occurred—is that *survival value and intellectual power are not the same thing*. That is why the genetical or evolutionary search, item (b) on the list, is a *different, distinct, variety of search* than either intellectual search (item (a) on the list) or cultural search (item (c) on the list). In intellectual search (item (a)), it is brains that carry out the search; in genetical or evolutionary search (item (b)) it is natural selection, or nature, carrying out the search (for a combination of genes").[5] And, in cultural search (item (c) on the list) it is "the human community as a whole" that carries out the search, rather than either a human (brain) or nature. Thus, these varieties of search can be distinguished by the *agent to whom Turing attributes the action of carrying out the search*, when the process is regarded as a search of that kind.

Turing notes that for a human to develop "intellectual power", he or she must "be immersed in an environment of other [humans], whose techniques he or she absorbs during the first twenty years of his [or her] life. He may then do a little research of his own and make a very few discoveries which are passed on to other [humans]." It is in this passage that he makes the distinction between the agency of the individual being trained and the agency of a human community of which the individual later

[4] Quoted in (Bengtsson and Tunlid 2010), p. 712.

[5] I think that Turing is very clear about the fact that he is drawing an analogy when he lays out the analogy between natural selection and the analogue he proposes be tried (in his later (1950) paper Computing machinery and intelligence). In the analogy in that paper, which he explicitly lays out there, he says that "the judgment of the experimenter" is analogous to Natural selection. In that analogy, "how well [the machine] learns" would be analogous to how well an animal form does in terms of survival.

becomes a part, writing that "from this point of view the search for new techniques must be regarded as carried out by the human community as a whole, rather than by individuals." (Turing, 2004a, p. 440) The importance of being immersed in an environment of other humans is emphasized at some other places in the (1948) "Intelligent Machinery" report for NPL, too. When pointing out that "although a [human] when concentrating may behave like a machine without interference, his behavior when concentrating is largely determined by the way he has been conditioned by previous interference", he notes not only that this interference includes "contact with human beings for twenty years or more", but also that a human is "in frequent communication with other [humans]." (Turing 2004a, p. 430).

14.4 Being in a Search Party

The capability to carry out intellectual search (for combinations with particular properties), then, is ascribed to a brain, and the development of an individual into a human with intellectual power is dependent not just on having a good human teacher, but on being surrounded by and interacting with lots and lots of other humans, over the course of many years, and continuing to do so throughout one's adult life. In fact, the kind of interaction that continues to be frequent is "*communication* with other [humans]" (italics added). I believe that it is this "frequent *communication*" aspect that is crucial to being able to make sense of Turing's ascription of the capability to make new discoveries to a human community, but not to an individual. He grants that individuals do conduct research and make discoveries, but he does not consider the "search for new techniques" to be something that is carried out by an individual human. Were the human interference with the machine limited to the educational phase of the machine (which occurs prior to the individual's research-and-discovery phase), it would not make a lot of sense to distinguish between the aspects of the individual's researches and discoveries that were to be ascribed to him as an individual, and the aspects that could only be seen as a contribution to a joint action carried out by "the human community as a whole." When, however, we consider the model of human researcher on which it is not just during an initial educational period that the researcher is in contact with others, but that he or she is "in frequent communication with other [humans]", it makes sense that, although some of the individual's research and discovery could be regarded as carried out by the individual, that he or she is also participating in a joint action of discovery of which the individual is not the agent carrying out the joint action of discovery.

Turing predicted that "Further research into intelligence of machinery will probably be very greatly concerned with 'searches' of [the intellectual] kind' " but the portion of his paper about "the cortex" should be seen as only part of the larger inquiry whose lines the paper sketches out. I think we can see where he means the activity of intellectual search fits into this sketch with respect to his comments on "the cortex" when he writes about the role of "definite nerve circuits required for

quite definite purposes": "all the reflexes proper (not 'conditioned') are due to the activities of these definite structures in the brain. Likewise the apparatus for the more elementary analysis of shapes and sounds probably comes into this category. But the more intellectual activities of the brain are too varied to be managed on this basis." (Turing 2004a, p. 432). I want to focus on how the issues brought out above bear on this observation. That is, how the significance that Turing placed on the human's advantage over a computer in being immersed in a community of humans, of having had a lot of contact with other humans over the course of many years, and, finally, of being in frequent communication with others bears on the observation that the more intellectual activities of the brain are too varied to be managed by definite structures in the brain. That means they are due to interference, or contact, with other things; Turing's view seems to be that to develop true intellectual power, that interference involves frequent communication with other humans. The question for a machine is: How much, and what kind of contact does a machine need in order to carry out various kinds of search well?

14.5 Human Help and the Human Brake

It is noteworthy that Turing cites the *value* of a machine's contact with (or interference from) humans—especially frequent communication with others—in order to gain "intellectual power" in this 1948 paper in which he is exploring how to educate a machine on analogy with how humans are educated so as to come to have the intellectual capabilities that they do is. Interaction with humans carried another association in other studies of the capabilities of computing machinery; in a 1945 investigation into building a computer, Turing had been concerned about the *negative* effects of involving humans in a computing machine's operations. Marie Hicks notes that Turing coined the term "human brake" and explains: "The human brake slowed computing processes by delegating actions to human operators that should ideally reside within the capability of a machine and its programs" (Hicks 2008).

In terms of speed, we don't want our machines to be bogged down by having to interact with humans; as Turing wrote in "Proposed electronic calculator" in 1945: "Once the human brake is removed the increase in speed is enormous." (Turing 2005, p. 371) At that time, he stressed the crucial importance of a large memory for information storage in order for the machine to be able to carry out complicated things; as for human interaction: "How can one expect a machine to do all this multitudinous variety of things? The answer is that we should consider the machine as doing something quite simple, namely carrying out orders given to it in a standard form which it is able to understand." (p. 372)

When it came time to consider the possibility of intelligence, there was the theoretical issue that exhibiting intelligence required permitting the machine to be fallible (since "if a machine is expected to be infallible, it cannot also be intelligent." (Turing 2004c, p. 402)) After dealing with that theoretical point, though, when it came to achieving intellectual development of a machine, interaction with humans

was almost always cited as necessary for achieving parity with human displays of intelligence, and he proposed concrete, practical examples of rudimentary interactions by which a machine could improve: playing games with humans. Copeland refers to the 1948 "Intelligent Machinery" report for the NPL as "The First Manifesto of Artificial Intelligence"[6] and we have seen how salient the points about interaction with humans are in that paper. In his 1947 "Lecture on the Automatic Computing Engine", too, Turing had stressed the same point we see in the 1948 "Intelligent Machinery" report about the crucial nature of *interaction of the machine with humans*. He makes it, explicitly, in the closing. Arguing from a comparison with how humans come to gain the intelligence they do, he argued from the observation that "A human mathematician has always undergone an extensive training" to "the machine must be allowed to have contact with human beings in order that it may adapt itself to their standards." In fact, he said there that "One must . . . not expect a machine to do a very great deal of building up of instruction tables on its own" (Turing 2004c, p. 403).

The suggestion that human cognitive abilities can not develop in isolation was not new with Turing. Lev Vygotsky's theories[7] about the role of social interaction in the development of an individual human's cognitive abilities were published, although it is hard to tell how well known they were at that time. Turing's views that the development of "intellectual power" depends upon contact with other human beings is in sympathy with Vygotsky's views, but Turing's suggestion that it was the best way to approach machine intelligence, as opposed to using principles and rules, went beyond Vygotsky's views, which were limited to humans.[8]

Turing was certainly right in predicting that much research into machine intelligence would be concerned with searches of the intellectual kind—but how has his view about the value of interactions with humans fared? Let us examine two recent projects: IBM's Question-answering machine, Watson, and Carnegie-Mellon's language learning machine, NELL.

14.6 NELL – "Reading" the Web

According to Tom Mitchell, "The idea of NELL ("Never Ending Language Learning") is to capture a style more like the on-going learning of humans." By "style", though, he meant that humans are continuously, rather than intermittently, learning. So, NELL operates continuously, 24/7, "reading" the Web; the overall goal is to acquire the ability to extract structured information from unstructured web pages. However, unlike Turing's idea of having a human teacher of the

[6] In Copeland (ed.) 2004, p. 409.

[7] Vygotsky"s *Thought and Language* was published in 1934, and he had traveled to London before that.

[8] In a suggestive paper (Lindblom and Ziemke 2003) Jessica Lindblom and Tom Ziemke discuss how Vygotsky's views Vygotsky (1986) might apply to designing human-robot interaction.

child-machine, NELL was left to operate autonomously for the first 6 months, after being given a list of several hundred categories and relations, and about "10–15 seed examples" of each.[9] What NELL is supposed to do is to form beliefs, and to determine how strongly to believe (what confidence level to assign to) them.

The approach taken is called "Macro-Reading"; the motivation for the idea is to tackle a problem that is much more tractable than the problem of natural language processing. Mitchell explains that "the problem of automatically populating large databases from the internet can be formulated so that it is much easier to solve than the problem of full natural language understanding." Macro-reading (in contrast to micro-reading) is defined "as a task where the input is a large text collection (e.g., the web) and the desired output is a large collection of facts expressed by the text collection, without requiring that every fact be extracted." This is supposed to be simpler because the redundancy of the facts on the web allows the reader to ignore complex statements and yet still pick up on the main facts available on the web. Other features of NELL that make the task of "reading" the web more tractable are that NELL aims for something rather streamlined rather than embracing all the bricolage surrounding a category or relation on the massive web. NELL doesn't try to understand the text it encounters. NELL's task is to "populate" a portion of the semantic web by "populating", i.e., filling in, an ontology that is given as input. Finally, NELL does figure out some ways to extract information from the text based on patterns; Mitchell's example is " 'mayor of X' often implies X is a city."[10]

NELL comes up with beliefs and these are listed on the web. Help from humans in scoring confidence levels for these "beliefs" is crowd-sourced; for each "belief" listed, a visitor to NELL's website can click a "thumbs up" or "thumbs down" to help NELL correct, or fine tune, the confidence level associated with that "belief." On August 27th, NELL learned that "belfast_zoo is a zoo in the city belfast" and that had a 100% confidence level. But "family_animals is a mammal" did not rise so high, nor did "savior is the parent of trusting jesus." The visitor cannot interact with NELL to edit, revise, or ask questions about, these beliefs; interaction is limited to a "correct" or "incorrect" response. I think it is clear that what NELL is going to gain from interacting with humans is only going to take it so far.[11]

With respect to our question, what I would say about NELL is that its creators aimed for a system that "learned" (in the sense of improving its performance in extracting true statements from web pages) autonomously, and because they were interested in making it as autonomous as possible, they were not thinking in terms of how best to incorporate interactions with humans into its learning process. As it turned out, human help was later added, both in the form of additional constraints on the relations NELL worked with, and also (via crowdsourcing) with confidence levels of the beliefs it generated. So, NELL does make some (slight) use of ongoing interaction with humans. However, it is not discriminative with respect to which

[9] "NELL: The Computer that Learns", downloaded 25 January 2014. http://www.cmu.edu Web.

[10] In (Mitchell et al. 2009).

[11] Checking NELL's "Recently-Learned Facts" on January 26, 2014, I find that "h_ross perot is a politician who holds the office of president" is held with 99.2% confidence!

humans' input it uses; this may account for its acquisitions of some strange "beliefs" that are scored with high confidence ratings from time to time.

In a deeper sense, though, I would say that NELL did not really take advantage of opportunities to learn from humans via the web. This failure is inherent in the "macro-reading" approach, since its modest aim is to extract some simple relations that appear frequently on the web (and, later, to mine its set of these relations for rules it can use to conclude more simple relations (Lao et al. 2011). In contrast, I find that IBM's Watson, in part because its task was different, was perfectly situated to learn from humans via the web, as well as via more direct kinds of "interference" (i.e., programming). As I'll explain, understanding how Watson made use of human contributions via the web makes Watson in some ways more impressive as a learner, but makes Watson's performance on the game show *Jeopardy!* somewhat less impressive than it appears if one is in a state of ignorance about how Watson achieved its success.

14.7 IBM's Question-Answering Champion, Watson

IBM's Watson is a question-answering system that was specially designed, constructed, and tested to compete in the television game show *Jeopardy!* It is very sophisticated. It won against the reigning human champions of the game. Watson uses a software architecture called DeepQA. It employs a variety of languages and methodologies, and it uses parallel computer architecture to provide the necessary speed. It performs an analysis of the question to determine what is being asked, then generates many candidate answers. Each candidate answer is checked against many sources for consistency and other kinds of checks, and, eventually, confidence levels are computed and candidate answers are ranked. Watson then uses these ranked candidate answers from DeepQA, along with other information it generates, to play the game (e.g., to choose clues, answer questions, decide how much to wager). The confidence levels are important to playing the game well, since wrong answers incur substantial penalty.

As answering the questions to *Jeopardy!* requires knowledge of specific details about a wide variety of publicly available information, Watson, too, was taught to use the unstructured information on the web. However, in complete contrast to NELL, Watson does not avoid natural language processing, nor look for simpler tasks such as "macro-reading."

The approach to making use of the vast amount of information on the web that was taken for building Watson was to be discriminating about the quality of the sources acquired, and to be practical about the amount of text involved in a source.

> A key element in high-performing question-answering (QA) systems is access to quality textual resources from which answers to questions can be hypothesized and evaluated. [. . .]
>
> We developed three procedures to obtain high-quality textual resources, i.e., source acquisition, source transformation, and source expansion. When developing a new QA system or adapting an existing system to a new domain, relevant sources need to be identified

> to cover the scope of the task. We refer to this process as *source acquisition*, which is an iterative development process of acquiring new collections of documents to cover salient topics deemed to be gaps in existing resources. The acquired sources are examined with respect to characteristics of system components, as well as to the nature of the questions and the answers in the new domain to ensure that they are represented in the most effective manner. Some acquired sources go through a process that we call *source transformation*, in which information is extracted from the sources, either as a whole or in part, and is represented in a form that the system can most easily use. Finally, whereas source acquisition helps ensure that the system has coverage in salient topics of the domain, *source expansion* attempts to increase the coverage of each known topic by adding new information, as well as lexical and syntactic variations of existing information. We believe that the methodology that we developed for source acquisition, transformation, and expansion is crucial for providing Watson with the necessary resources to achieve high QA performance." (Chu-Carroll et al. 2012a, pgs. 4:1–4:2)

Commentators on Watson's performance often make much of the fact that the *Jeopardy*! game requires knowledge about a very wide range of topics in a very wide range of disciplines: history, literature, sports, popular culture, science, mathematics, geography, politics, and so on. It wasn't just that Watson was able to outperform humans in a question-answering task, but that Watson was able to outperform humans in a question-answering task that was (seemingly) unrestricted with respect to topic, that was so remarkable.

However, Watson's designers share something discovered in the course of source acquisition that changes one's perspective on this feat a bit: "Fortunately, although the questions cover many domains, for the most part, they cover popular topics within those domains and represent information of interest to the general audience." (Chu-Carroll et al. 2012a, p. 4:2) Investigating further, they found something astounding: "on a randomly selected set of 3500 questions, all but 4.53% of the answers were Wikipedia titles."[12] What to make of this? It didn't really matter what this said about the game show or about Wikipedia, it was a useful conclusion that made the task of building Watson much easier. Wikipedia article titles "can serve as an excellent resource for candidate-generation." (Chu-Carroll et al. 2012b, 6:8) Wikipedia articles provided useful metadata, too. Metadata (such as what text in the article links elsewhere, and what it links to) helped determine what was salient to a topic and what was not. Watson's designers found "We observed that plausible candidates typically satisfy two criteria. First, they represent salient concepts in the passage. Second, the candidates have Wikipedia articles about them." (Chu-Carroll et al. 2012b, 6:8)

By itself, just using the corpus of Wikipedia to generate candidate answers wasn't enough. But Wikipedia articles could help with more than just candidate generation. They may count as unstructured text, but there is a lot more than simple text in them. Chu-Carroll et al. 2012a explains how the team was able to make use of the "title-oriented" character of Wikipedia entries. Other articles (Chu-Carroll

[12] Of the 4.53% of answers that were not Wikipedia titles, "some are multiple answer questions (e.g., "Indiana, Wisconsin, and Ohio" and "heat and electricity"), some are synthesized answers to puzzle questions (e.g. "TGIF Murray Abrahams" and "level devil") and a small number are verb phrases (e.g., "get your dog to heel") (Chu-Carroll 2012a; 4:2)

et al. 2012c, p. 12:3) show how the links in Wikipedia articles provide help in identifying implicit relationships, to help Watson build up its store of knowledge about concepts that were closely related, which can come in handy when figuring out what a clue is asking for.

This leg-up in generating candidate answers had numerous benefits, but there is one that deserves special mention: a better way to handle assigning a type to the answer that is sought by a particular question. The usual order of things in state of the art question-answering programs of the time was to first figure out what "answer type" the answer to the question would have:

> Many open-domain question-answering (QA) systems adopt a *type-and-generate* approach by analyzing incoming questions for the expected *answer type*, mapping it into a fixed set of known types, and restricting candidate answers retrieved from the corpus to those that match this answer type (using type-specific recognizers to identify the candidates.) (Murdock et al. 2012, p. 7:1)

The Watson team switched the order of things, using instead a *generate-and-type* framework. Types matter, but checking that the type of the candidate answer fits what the question asked is done much later, as part of the task of assigning confidence scores to each answer and ranking them according to confidence level. This permitted many more types, and it allowed for much more flexibility about types. The designers explain that early on, in analyzing the domain of questions from the TV quiz show *Jeopardy!*, they found the type-and-generate approach "to be problematic." They found, in fact, that they could not reliably predict "what types the questions will ask about and what their instances are." The number of type words that could occur in a *Jeopardy!* style question was, practically speaking, at least, unbounded. A survey of 20,000 questions yielded "roughly 5000 different type words." (Murdock et al. 2012, p. 7:3) In explanation of this striking situation, they give some examples:

- Human language is remarkably rich when it comes to assigning types; nearly any word can be used as a type, particularly in some questions.
- Invented in the 1500s to speed up the game, this *maneuver* involves two pieces of the same color. (Answer: "Castling")
- The first known airmail service took place in Paris in 1870 by this *conveyance.* (Answer: 'hot-air balloon")
- In 2003, this Oriole first *sacker* was elected to the Baseball Hall of Fame. (Answer: "Eddie Murray")

An answer type, they concluded, ought to be treated as a property of the question and answer combined. The approach they developed differed from the usual QA systems approach of looking for candidate answers of the right type; instead, it was to "find candidates (in some way) and judge whether each one is of the right type by examining it in context with the answer type from the question. " (Murdock et al. 2012, p. 7:3)

I suspect this approach, which was in part possible because of the leg up on generating candidate answers provided by Wikipedia entries, was important to Watson's stellar showing in a game that required not just a basic competence in natural language processing, but lots of savvy about that language.

Isn't Wikipedia really just another unstructured text source, though? Yes, it is an unstructured text source, but I also think that Wikipedia is special. Wikipedia entries are written to be read by someone with a question, and Wikipedia text is constructed by humans who enjoyed writing and sharing the information—and were joined by others who added to and revised it. The article titles are constructed for the web visitor looking for an answer, and the information, though unstructured text, is still organized in predictable ways (i.e., the links it contains will not be arbitrary words in the entry, but will be especially relevant to the title topic.) It is generally targeted towards the general, i.e., non-expert reader. And, not only is it comprehensive in time backwards, but it is constantly being updated, so that it can always be up to date—on everything it covers. In some ways, having access to Wikipedia articles on a continuous basis resembles being in frequent communication with humans.

I think it speaks well of the Watson team that their methodology quickly uncovered the relationship between Wikipedia entries and Jeopardy! answers, and that their approach makes such rich use of the various aspects of the Wikipedia entries. Were the Watson technology to be adapted to another use, i.e., as a collaborator in some other professional field, the kinds of documents that are important might be, in fact probably will be, different ones. More generally, Watson receives training from humans, who tell it which sources to prefer over others. One of the many things the humans told Watson for its Jeopardy! contestant "job" was to use Wikipedia and how to use it. Watson still generates candidate answers and metadata on its own, still performs lots of checks and evaluations on candidate answers, and still comes up with confidence rankings on its own, but a good part of the reason it does all these things so well is the help from humans it receives in terms of getting quality sources and being told how to make good use of them. Thus it seems to me that the case of Watson bears Turing out; the machines that are closest to developing intellectual power on parity with humans are those that are not only trained by humans, but that are in frequent communication with them.

14.8 Closing Thoughts

Both NELL and Watson have access to the same unstructured text on the web. If Watson (when connected to the internet) can be seen as having interaction with humans that approximates communication with them via the web, why not NELL, too? Well, communication takes two. The way NELL learns from Wikipedia pages is very different from the way Watson does. NELL sees the text on the web, not as text to be processed in order to understand what its author meant to communicate, but as a landscape studded with a plethora of simple facts caught in a complicated matrix of other text. The landscape is to be mined by ignoring all of the complication in that complicated matrix, and paying only scant attention to the context surrounding the simple fact. Watson, on the other hand, sees the text on the web as much more variegated; it uses only quality text sources and (via an iterative process) ensures it has enough of the right kinds of them to address its anticipated needs. The

approach Watson's designers took appreciates that the text is authored by someone who meant to communicate something, and so designed Watson to analyze the text accordingly, not only to understand what its author meant to communicate, but also to get the most information out of the text that it can. That means, loosely speaking, that Watson attempts to understand the text as natural language, of course, but also that Watson uses other features about the text (e.g., which string of text is designated as the title, which strings of text are anchors for weblinks) that yield information on how things and concepts may be related, and that Watson uses metadata drawn from links and statistical data.

We may suffer a bit of disillusionment upon finding out just how important the work of all those human Wikipedia contributors was in Watson's *Jeopardy!* win, but, in a sense, Watson is impressive for knowing how to make such good use of what the Wikipedia contributors provided.On this view of things, it is Watson whose approach might someday lead to real intellectual power, and for reasons akin to Turing's emphasis on the importance of human contact: Watson knew how to listen to and learn from what humans were telling it.

References

Bengtsson, B. O., and A. Tunlid. 2010. The 1948 International Congress of Genetics in Sweden: People and Politics. *Genetics* 185: 709–715. Also available at http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2907196

Chu-Carroll, J., J. Fan, N. Schlaefer, and W. Zadrozny. 2012a. Textual Resource Acquisition and Engineering. *IBM Journal of Research and Development* 56(¾), Paper 4 May/July 2012.

Chu-Carroll, J. et al. 2012b. Finding Needles in the Haystack: Search and Candidate Generation. *IBM Journal of Research and Development* 56(¾), Paper 6 May/July 2012.

Chu-Carroll, J., E. W. Brown, A. Lally, and J. W. Murdock.2012c. Identifying Implicit Relationships. *IBM Journal of Research and Development* 56(¾) Paper 12 May/July 2012.

Copeland, J.B., ed. 2004. *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life, Plus the Secrets of Enigma*. New York: Oxford University Press.

Hicks, M. 2008. Repurposing Turing's 'Human Brake'. *Annals of the History of Computing.* 30(4).

Koza, J. R., F. H. Bennett III, D. Andre, and M. A. Keane. 1999. Genetic Programming: Turing's Third Way to Achieve Machine Intelligence. EUROGEN workshop in Jyvdskyld, Finland on May 30 – June 3, 1999. Available at http://www.genetic-programming.com/jkpdf/eurogen-1999turing.pdf.

Lao, N., T. Mitchell, and W. W. Cohen. 2011. Random Walk Inference and Learning in a Large Scale Knowledge Base. In EMNLP '11 Proceedings of the Conference on Empirical Methods in Matural Language Processing, 529–539. Stroudsburg: Association for Computational Linguistics.

Lindblom, J., and Y. Ziemke. 2003. Social Situatedness of Natural and Artificial Intelligence: Vygotsky and Beyond. *Adaptive Behavior* 11 (2): 79–96.

Mitchell, T. M., J. Betteridge, A. Carlson, E. Hruschka, and R. Wang. 2009. Populating the Semantic Web by Macro-Reading Internet Text. In *Proceedings of the International Semantic Web Conference* (ISWC).

Murdock, J. W. et al. 2012. Typing Candidate Answers Using Type Coercion. *IBM Journal of Research and Development* 56(¾), Paper 7 May/July 2012.

NELL: The Computer that Learns. http://www.cmu.edu/homepage/computing/2010/fall/nell-computer-that-learns.shtml. Accessed 25 Jan 2014

Sterrett, S. G. 2012. Bringing up Turing's 'Child-Machine'. In *How the World Computes: Turing Centenary Conference and 8th Conference on Computability in Europe, CiE 2012, Cambridge, UK, June 18–23, 2012. Proceedings*, Lecture Notes in Computer Science, Vol. 7318, ed. S. Barry Cooper, Anuj Dawar, and Benedikt Löwe, 703–713, Berlin/Heidelberg.

Turing, A.M. 2005. Proposed Electronic Calculator (1945). In *Alan Turing's Automatic Computing Engine: The MASTER Codebreaker's Struggle to Build the Modern Computer*, ed. B.J. Copeland, 369–454. Oxford/New York: Oxford University Press.

Turing, A. M. 2004a. Intelligent Machinery (1948). In ed. J. B. Copeland, 418–440.

———. 2004b. Computing Machinery and Intelligence. In ed. J. B. Copeland, 418–440.

———. 2004c. Lecture on the Automatic Computing Engine. In ed. J. B. Copeland, 378–394. New York: Oxford.

Vygotsky, L. S. 1986. *Thought and Language*. Cambridge, MA: MIT Press (Original work published 1934).

Chapter 15
Is There a Church-Turing Thesis for Social Algorithms?

Rohit Parikh

Abstract It is well known that how an individual acts in a specific situation depends not only on her preferences (and her means) but also on what she believes. If she believes a restaurant is open, she will go to it, assuming she likes that restaurant and she has the means, e.g., transportation. But if she knows it is not open, then her liking becomes irrelevant.

When someone wants some behavior to come about on a large scale he must arrange for the various individuals involved to have the information which they will need to carry out their own role in that behavior. If you put on a concert, you must also advertise.

In some cases, if the interests of the individuals diverge from that of the planner, then he may also try to see to it that they lack certain information. No one wants his or her social security number to be common knowledge but everyone wants to know when the bus is coming. So the dance between knowledge and ignorance drives much social action.

We will offer a formal account of the way this happens, using tools from epistemic logic, game theory and the analysis of algorithms.

The simple answer to the question raised in the title is:

No (This is an expository paper containing some recent ideas. (Pacuit E, Parikh R, Social interaction, knowledge, and social software. In: Goldin D, Smolka S, Wegner P (eds) Interactive computation: the new paradigm. Springer, New York, 2006) contains a nice survey. The results on campaigning are joint with Walter Dean (Dean W, Parikh R, The logic of campaigning. In Presented at logic and its applications, lecture notes in computer science, Volume 6521, 4th Indian Conference, ICLA 2011, Delhi, India, January 5–11, 2011). The material on influencing action by influencing knowledge is joint with Çağıl Taşdemír and Andreas Witzel (Int Game Theory Rev 15:1–28, 2013)).

R. Parikh (✉)
City University of New York, Brooklyn College and CUNY Graduate Center,
New York, NY, USA
e-mail: RParikh@gc.cuny.edu; http://www.sci.brooklyn.cuny.edu/cis/parikh/

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6_15

Turing was dealing with a relatively simple situation where he was interested in algorithms which took in numbers (or strings) as input and producing an output. Social algorithms deal with much more complex 'inputs' and the 'output' may be something more complex as well. And yet the project of developing a taxonomy of social algorithms seems both important and interesting.

15.1 Two Ways for Social Algorithms

15.1.1 Individual Level

Sen (1999) points out in *Commodities and Capabilities* that our reliance on utility theory is a bit naive, for people want many things which may not be comparable:

> I argue in favor of focusing on the capability to function, i.e., what a person can *do* or can *be* and argue against the more standard concentration on *opulence* (as in 'real income' estimates) or on *utility* (as in traditional 'welfare economic' formulations). Insofar as opulence and utility have roles (and they certainly do), these can be seen in terms of their indirect connections with well-being and advantage, in particular, (1) the *causal* importance of opulence, and (2) the *evidential* importance of utility (in its various forms, such as happiness, desire-fulfillment…

Suppose that I am going from my apartment to a hotel in Chicago. Then my schedule (or program) will consist of several steps. Taking a cab to LaGuardia airport, checking in, boarding the plane, and then in Chicago, taking a cab to the hotel. Each step needs to be tested for correctness. For instance, if I do not have a picture ID, I cannot board the plane, and it is irrelevant that the other steps would have gone well.

But, what I can do depends on resources which society has provided. A societal investment in an airport or a public bus improves my capabilities and therefore my wellbeing without any actual income-transfer taking place from society to (indigent) individuals.

The schedule I described is a simple *straight line* program. There could also be decision points like, *take a taxi if available, otherwise take a bus.* A plan with many such decision points would look like a tree rather than a line. But the entire schedule does have to be checked for correctness—something we do informally.

But other social procedures like organizing a conference or a wedding can be far more complex. I can invite aunt Betsy only if her ex-husband Eric has said he cannot come.

One role which society can play in our lives is to serve as an *operating system* within which we can write our individual programs. When society builds a bridge or starts a bus line it increases our capabilities and thereby increases our wellbeing.[1]

[1] See (Dean and Parikh 2011) for an analysis of computer algorithms, which can serve as a very rough template for our concern here.

15.1.2 The Social Level

> The peculiar character of the problem of a rational economic order is determined precisely by the fact that the knowledge of the circumstances of which we must make use never exists in concentrated or integrated form, but solely as the dispersed bits of incomplete and frequently contradictory knowledge which all the separate individuals possess. The economic problem of society is thus not merely a problem of how to allocate "given" resources-if "given" is taken to mean given to a single mind which deliberately solves the problem set by these "data:" It is rather a problem of how to secure the best use of resources known to any of the members of society, for ends whose relative importance only these individuals know. (Hayek 1945).

Sometimes a social planner wants to achieve some social goal.

He needs to be aware that the plan can only succeed if the various individuals act in a manner which is consonant with the plan and that they will act in a certain way depending on their beliefs, on their desires and on their possibilities of action.

So their actions can be influenced either by influencing their beliefs or their possibilities of action or their desires (the last is harder but can be achieved via advertising or brain washing).

Michael Chwe (2001) points out that royal processions have this effect (and probably the purpose) of projecting the king's power and making rebellion seem futile.[2]

15.2 Knowledge in Action

15.2.1 Example: Knowledge in a Restaurant

Three people A, B, C walk into a coffee shop. One of them orders cappuccino, one orders tea, and one orders ice cream. The waiter goes away and after 10 min *another* waiter arrives with three cups. "Who has the cappuccino?" "I do," says A. "Who has the tea?" "I do," says C.

Will the Waiter Ask a Third Question?

Consider the possible situations for waiter 2.

1) CTI 2) CIT
3) TCI 4) TIC
5) ICT 6) ITC

[2] Sometimes the social planner, in order to carry out his plans, needs to know about the preferences and beliefs of the individuals affected by his plans. In some cases he can work out a plan where individuals act honestly, the Vickrey auction (1961) is an example of such a device. But the Gibbard-Satterthwaite theorem tells us that such devices are not generally available and strategizing in voting behavior cannot be avoided, see (Brams 2007).

When A says that he has the cappuccino, 3,4,5,6 are eliminated. The waiter now has

1) CTI 2) CIT

When C says that he has the tea, 1 is eliminated, leaving only 2.
2) CIT.

Now the waiter knows that B has the ice cream.

15.2.2 Another Example

A butler about to clean a hotel room comes across a lady guest, coming out of the bathtub and not even wearing a towel.

"Excuse me, sir," says the butler and withdraws.

Why did the butler say "sir"?

Because in the lady's mind there were two possibilities.

1. The butler saw her clearly.
2. The butler did not see her clearly.When the butler said "sir", he eliminated 1. and saved the lady from embarrassment.[3]

15.2.3 Inducing Beliefs: Shakespeare's Much Ado About Nothing

At Messina, a messenger brings news that Don Pedro, a Spanish prince from Aragon, and his officers, Claudio and Benedick, have returned from a successful battle. Leonato, the governor of Messina, welcomes the messenger and announces that Don Pedro and his men will stay for a month.

Beatrice, Leonato's niece, asks the messenger about Benedick, and makes sarcastic remarks about his ineptitude as a soldier. Leonato explains that "There is a kind of merry war betwixt Signior Benedick and her."

[3] A political leader, say Obama, will use such a device to avoid embarrassment. He may ask some member of his cabinet to announce a policy being considered. If the policy proves to be popular, the president can then go ahead and say that the announcement was made with his approval. If it proves to be unpopular, then the president can say that cabinet member spoke prematurely and that the actual policy is different.

Robert Stalnaker (2005) considers a case where the secretary of the treasury John Snow makes an announcement that when the dollar is low it improves exports. This announcement did indeed lower the dollar and there was some perplexity whether the announcement was deliberate so as to improve exports, or whether it was simply a mistake on the part of Snow who did not realize the power of his words.

Table 15.1 Benedick's decision problem

	Love	*Nolove*
Propose	100	−20
Nopropose	−10	0

Various events take place and Claudio wins the hand in marriage of Hero, Leonato's only daughter and the wedding is to take place in a week.

Don Pedro and his men, bored at the prospect of waiting a week for the wedding, hatch a plan to matchmake between Beatrice and Benedick who inwardly love each other but outwardly display contempt for each other.

According to this stratagem, the men led by Don Pedro proclaim Beatrice's love for Benedick while knowing he is eavesdropping on their conversation.

Using $K_i(x)$ for i knows the proposition x, b for Benedick, d for Don Pedro and E for the event of eavesdropping, we then have

$$K_b(E), K_d(E) and \neg K_b(K_d(E))$$

All these conditions are essential and of course the plot would be spoiled if we had $K_b(K_d(E))$ instead of $\neg(K_b(K_d(E))$. Benedick would be suspicious and would not credit the conversation.

The women led by Hero carry on a similar charade for Beatrice. Beatrice and Benedick are now each convinced that their own love is returned, and hence decide to requite the love of the other (Table 15.1).

The play ends with all four lovers getting married.

Here *love* means "Beatrice loves me" and *nolove* the other possibility. If love is true, then *propose* is better than *nopropose*; and if *nolove* is true, i.e. love is false, then *nopropose* is better. By getting Benedick to believe love, Don Pedro changes Benedick's best move.

15.3 Formalism

We create a language to talk about various knowledge properties in the following way.

- An atomic predicate P is a formula
- If A, B are formulas, then so are $\neg A$ and $A \wedge B$
- If A is a formula and i is an agent, then $K_i(A)$ is a formula
- We may also include formulas $C(A)$ if we wish to denote common knowledge.

15.3.1 Intuition

Intuitively $K_i(A$ means that the agent i knows the fact expressed by the formula A. $K_jK_i(A)$ means that j knows that i knows A.

If i,j are the only agents, then $C(A)$ means that i knows A, j knows that i knows A, j knows that j knows that j knows A, and so on forever.

Example:
For example, suppose Ravi and Usha are playing cards. There is a mirror behind Usha so Ravi can see her cards. But Usha does not know this.

And there is a mirror behind Ravi, and Usha can see Ravi's cards. But Ravi only knows about Usha's mirror. And Usha only knows about Ravi's mirror.

Suppose Ravi has the Queen of spades. Let Q represent this fact. Then we have $K_r(Q)$ of course, $K_u(Q)$, and naturally $K_uK_r(Q)$.

But we do not have $K_rK_u(Q)$.

15.3.2 Kripke Structures

Kripke structures are used to interpret the language above.

Kripke structure M for knowledge for n knowers consists of a space W of states and for each knower i a relation $R_i \subseteq W \times W$.

There is a map π from $W \times A \to \{0, 1\}$ which decides the truth value of atomic formulas at each state.

Usually the R_i are taken to be equivalence relations, i.e., reflexive, symmetric and transitive.

We now define the truth values of formulas as follows:

1. $M, w \vDash P$ *iff* $\pi(w, P) = 1$
2. $M, w \vDash \neg A$ *iff* $M, w \nvDash A$
3. $M, w \vDash A \wedge B$ *iff* $M, w \vDash A$ and $M, w \vDash B$
4. $M, w \vDash K_i(A)$ *iff* $(\forall t)(wR_it \to M, t \vDash A)$

$K_i(A)$ holds at w (i knows A at w) *iff* A holds at all states t which are R_i accessible from w.

15.3.3 Some Consequences

Since R_i is reflexive then we will get $K_i(A) \to A$ (veridicality) as a consequence.

Moreover, regardless of the properties of R_i we have.

1. If A is logically valid, then A is known
2. If A and $A \to B$ are known, then so is B

This is the well-known problem of *Logical Omniscience* since we are attributing knowledge properties to agents which they do not actually have.

Still, in *small* settings, such assumptions are reasonable.

15.3.4 Axiom System

1. All tautologies of the propositional calculus
2. $k_i(A \to B) \to (k_i(A) \to K_i(B))$
3. $K_i(A) \to A$
4. $K_i(A) \to K_iK_i(A)$
5. $\neg K_i(A) \to K_i(\neg K_i(A))$

Some of these axioms are controversial but we will not discuss the controversy.

There are also two rules of inference. *Modus Ponens*, to infer B from A and $A \to B$. And the other is *generalization*, to infer $K_i(A)$ from A.

The second rule does *not* say that if A is true than i knows it. Only that if A is a logical truth (or a logical theorem) then i knows it.

These rules are complete. All valid formulas are provable using the axioms and rules.

15.3.5 Revising Kripke Structures When an Announcement Is Made

Suppose we are given a Kripke structure M. Then some formula φ is announced publicly.

The new Kripke structure is then obtained by deleting all states in M where φ did not hold.

15.4 Theory of Mind

A group of children are told the following story:

Maxi goes out shopping with his mother and when they come back, Maxi helps mother put away the groceries, which include chocolate. There are two cupboards, red and blue. Maxi puts the chocolate in the red cupboard and goes out to play.

While Maxi is gone, mother takes the chocolate out of the red cupboard, uses some of it to bake a cake, and then puts the rest in the blue cupboard.

Now Maxi comes back from play and wants the chocolate.

Where will Maxi look for the chocolate?

Children at the age of five or more say, *in the red cupboard.*

But children up to the age of three or four say, *Maxi will look in the blue cupboard.*

What 3 year old children lack, according to psychologists Premack and Woodruff (1978) is a *Theory of Mind.*

It is not clear that all adults have the theory of mind. Suppose Jill has just poured herself a cup of coffee and is looking around the table. Both the milk and the sugar are next to Jack. Jack *may or may not* automatically push the milk and the sugar towards Jill. If he fails in this act of politeness then we might want to say that he is lacking in some aspect of the theory of mind. At the same time, there is little doubt that, speaking generally, humans (even children) have much more of a theory of mind than members of other primate species.

15.4.1 Animal Cognition: Do Animals Have a Theory of Mind?

Despite the more developed theory of mind we hold to be characteristic of humans as opposed to other primates, there are many questions about what is involved in having a theory of mind at all: which capacities, to what degree, and for which tasks. We might say that other primates *do* hold a theory of mind to some extent, as the following discussion shows.

15.4.2 Inducing False Beliefs in the Tigers of the Sundarbans

The Sundarbans are an area at the border of India and Bangladesh where lush forests grow and tiger attacks on humans have been common.

Fishermen and bushmen then created masks made to look like faces to wear on the back of their heads because tigers always attack from behind. This worked for a short time.

In 1987 no one wearing a mask was killed by a tiger, but 29 people without masks were killed. Unfortunately the tigers eventually realized it was a hoax, and the attacks resumed.

In Fig. 15.1 the chimp at the bottom is subservient to the dominant chimp at the top and has to decide which group of bananas to go for. In experiments, the sub-chimp tends to go for Food 1 which the dom-chimp cannot see. Is there use of epistemic logic by the sub-chimp? This is an issue of some controversy.

Fig. 15.1 What chimps think about other chimps

15.4.3 Knowledge Leads to Action

So far a great deal of work has been done on reasoning about knowledge, studying how knowledge is transferred and updated, and even some paradoxical theories of knowledge.

But we do not acquire knowledge, or transmit it to others just to pass the time.

Knowledge Means Influence and Power

It is a commonplace that what we do depends on what we know. And given that most of us have at least the rudiments of a theory of mind (cf. (Premack and Woodruff 1978)) we also know that what others do will depend on what *they* know.

15.4.4 Is Knowledge Always Beneficial?

Kamien et al. (1990) consider the following example. A black or white card is chosen from a deck and player 1 is invited to guess its color. After 1 makes her choice, which is announced, player 2 is invited to make a choice. The payoffs are as follows:

- If both players guess correctly, then both get 2
- If neither player guesses correctly, then both get 0
- If only one player guesses correctly, then the correct player gets 5 and the other player gets 0
- Suppose neither player knows the color; then player 1 should choose randomly, player 2 should choose a different color and the expected payoff for both is 2.5 (half of 5 + 0)
- If player 1 is allowed to see the card, then the dominant strategy for her is to announce the correct color, player 2 should choose the same color and the expected payoff for both (a certain payoff in fact) is 2

So the knowledge of player 1 makes her worse off.

However, player 1 is not harmed by the fact that she knows the color but by the fact that player 2 knows that 1 knows the color.

Neyman (1991) showed that if we can make one player know more but prevent other players from having more knowledge, then the player who knows more cannot lose.

We now consider how the same game may be played differently depending on the information available to the agents and their temperaments.

Suppose there is a *knowledge manipulator* —KM—who can control how much information the various agents can have then that agent can influence the way the game is played and the outcome.

Wife and Husband

We consider various scenarios involving the husband's knowledge and temperament. We assume that the wife knows the husband's payoffs and temperament and he does not know her temperament, though he knows her payoffs and knows that she expects him to move correctly (Fig. 15.2).

Case 1) Husband does not know wife's move (and she knows this).

(a) He is *aggressive*. Then being aggressive, he will choose S (Stravinsky) for his move since the highest possible payoff is 3. Anticipating his move, she will also choose S, and they will end up with payoffs of (2,3).
(b) If the husband is *conservative*, then not knowing what his wife chose, he will choose B since the minimum payoff of 1 is better than the minimum payoff of 0. Anticipating this, the wife will also choose B and they will end up with (3,2).

Case 2) If the husband *will* know the wife's move (i.e., what node he will be at), then the wife will choose B, the husband will also choose B and they will end up at (3,2).

We consider now the question of how the knowledge manipulator KM can create these various knowledge scenarios for the last example.

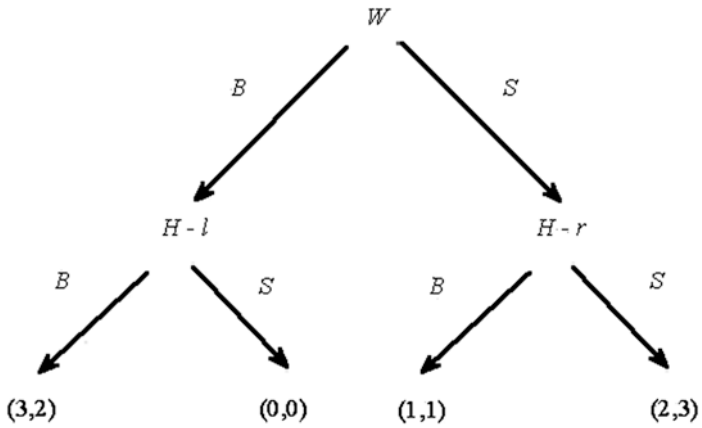


Fig. 15.2 We assume that the wife moves first and the husband after. "B" will represent the choice of Bach, "S" that of Stravinsky. "$H-l$" and "$H-r$" refer to left- and right-hand moves at nodes of the graph of possible moves

KM is capable of creating all these three situations by means of signals, as well as the one we did not mention where the husband does not know but the wife does not know that he will not.

Below, s is the function which, at a particular node of the tree, tells the KM what options s/he has for sending signals to wife and husband. Here the signals are l ("left"), r ("right"), and a neutral a ("indifference"). So, roughly, "s" stands for a signaling function. "s(H$-l$)" says, for example, what pairs of signals KM is allowed to send when the wife, husband pair are at H$-l$.

Husband and wife both know the function s, but when they do not know the argument (the node), they do not know the KM's options. And when KM has more than one option, they might not know which option KM actually chooses.

For case 1a),

$s(H - l) = (l, a)$ and $s(H - r) = (r, a)$.

If the situation is H$-l$, then wife gets an l and the husband gets an a. If it is H$-r$ then wife gets r, husband gets a again.

Suppose the wife gets an l. Since she got an l, she knows that they must be at node H$-l$, for that is the only node where KM can send an l to her, and here the husband has got an a. She also knows that he would have got an a also at H$-r$. So he does not know which node they are at, and she knows that he does not know.

The husband on the other hand gets an a and he knows that his wife got an l or an r.

But he does not know which. He knows that she knows he got an a in either case and so he knows that she knows that he does not know which node they are at.

For case 2),

$s(H - l) = (l, l)$ and $s(H - r) = (r, r)$.

Both will know which node they are at. In this case, if they are at H$-l$, they both get an l and so it is common knowledge that they are at H$-l$.

Finally, if KM wants the wife to be in doubt whether the husband knows, he could make.

$s(H - l) = \{(l, l), (l, a)\}$ and $s(H - r) = \{(r, r), (r, a)\}$.

Suppose they are at H−l. KM could send (l, l) or (l, a).

Suppose the first. He sends an l to both.

The wife knows that she got an l and so they are at H−l. But what did her husband get? He might have got an a and does not know where they are, or he might have got an l and does know where they are. Since he got an l he knows where they are but also knows that his wife does not know that he knows.

Suppose the second, and KM sends an l to wife and a to husband. The wife's knowledge is exactly as before. She knows where they are and does not know whether her husband knows. The husband does not know where they are (he could have got an a in either case) but also knows that his wife does not know that he does not know – for all she knows he might have got an l.

15.4.5 Learning from Communication

Observation 1 (Lewis (1969), Aumann (1976)): Suppose a group of people are commonly aware of a number of possibilities (states) among which they are uncertain. They commonly know some fact ψ if ψ is true of all these possibilities.

Observation 2 If a public announcement of some true formula φ is made, then the new situation is obtained by deleting all states s where φ is false.

As a candidate speaks, her position becomes more specific, and the number of possible actions she may carry out, becomes smaller.

What are the consequences for the candidate of this fact?

15.5 Brief Survey of 2008 US Election

- Major parties: *Democrats, Republicans*
- Main candidates for Democratic party: *Hillary Clinton, Barack Obama*
- Republican candidate: *John McCain*
- Winner of Democratic primary: *Barack Obama*
- Winner of election: *Barack Obama*
- Important political liability for McCain: *Iraq war* started under *George Bush*, a Republican president
- Important political liability for Obama: His association with *Jeremiah Wright*, a fiery black preacher who had made anti-America comments.

15.5.1 The Use of Language in Campaigning

The Onion, 1 May 2008:[4]

> After Barack Obama's comments last week about what he typically eats for dinner were criticized by Hillary Clinton as being offensive to both herself and the American voters, the number of acceptable phrases presidential candidates can now say is officially down to four. "At the beginning of 2007 there were 38 things candidates could mention in public that wouldn't be considered damaging to their campaigns, but now they are mostly limited to 'Thank you all for coming,' and 'God bless America'" [said] George Stephanopoulos.

15.5.2 An Illustrative Example

Hillary Clinton (while campaigning in Indiana):
D = As a child, I shot a duck.
Why would she say that?

- Indiana is a conservative state. So most of her immediate audience V_1 will be conservatives.
- Conservatives tend to *dis*favor gun control.
- Hearing D is likely to improve Hillary Clinton in the eyes of V_1, say by about amount u_1.
- But (virtually) all statements a candidate makes are **public announcements.**
- So another group of voters V_2 (say liberals in Massachusetts) also hear Hillary Clinton say D.
- This is likely to make her go down for V_2, say by amount u_2.
- But we likely have $|u_1| > |u_2|$ since

  (i) $|V_1| > |V_2|$, or at least V_1 cares more passionately about the issue than V_2.
  (ii) D merely *implicates* that Hillary Clinton will not impose gun control.

15.6 Towards a Formal Model: Languages and Theories

We begin by considering a single candidate C.

C*'s* views about the issues are formulated in a proposition language L containing **finitely many** atomic propositions $At = \{P_1, \ldots, P_n\}$.

For instance:

- P_1 = `We should withdraw from Iraq.`
- P_2 = `I will impose no new taxes.`
- P_n = `We should bail out the banks.`

[4] *The Onion* is a satirical weekly newspaper which comes out once a week in US cities.

T_a = C's **actual theory** (i.e. the entirety of her views)

T_c = C's **current theory** (i.e. what's she's said thus far)

Typically (but not always) $T_c \subseteq T_a$.

Given the theory generated by a candidate's statements, there is a set of possible worlds which are all compatible with that theory.

Let us call that set X_c. Then $X_c = \{w | \ w \vDash T_c\}$.

X_c is the set of those worlds which are compatible with what the candidate has said.

15.6.1 Worlds and Preferences

We conflate propositional valuations and worlds $w \in 2^{At}$.

We also define $w[i] = \begin{cases} 1 & \text{if } w \vDash P_i \\ -1 & \text{if } w \nvDash P_i \end{cases}$.

We initially consider a single group of voters V (think of this as a constituency).

The voters in V are characterized by their preference for an ***ideal world***.

This is formalized *via* two functions p_v , x_v:

- $p_v(i) = 1 \begin{cases} 1 & V \text{ would prefer } P_i \textit{ to be} \text{ true} \\ 0 & V \text{ is neutral about } P_i \\ -1 & V \text{ would prefer } P_i \end{cases}$
- $x_v : At \to [0, 1]$, the **weight** which V assigns to P_i such that $\sum w(i) \leq 1$.

15.6.2 Utilities of Worlds and Theories

The utility of a world for V is defined as

$$u(w) = \sum_{1 \leq i \leq n} p_v(i) \cdot x_v(i) \cdot w[i]$$

- Note that a candidate's current theory T_c is likely to be **incomplete**—i.e., she may not express a view on some P_i
- To calculate the utility of an arbitrary T we need to know how V will "fill in the blanks".
- That is, extend the evaluation from a single world to a **set** of worlds.

15.7 Voter Types

We postulate that there are three types of voters:

- **Optimistic voters** (assume the best about C given T_c).
- **Pessimistic voters** (assume the worst about C given T_c).

- **Expected value voters** (average across possibilities compatible with T_c).

  We will use a flat probability distribution, but only to simply our treatment.
  How will the different kinds of voters evaluate the candidate's theory T?

15.7.1 Voter Types

- Optimistic voters: $ut^o(X) = \max\ \{u(w) : w \vDash T\}$
- Pessimistic voters: $ut^p(T) = \min\ \{u(w) : w \vDash T\}$
- Expected value voters: $ut^e(T) = \dfrac{\sum_{w \vDash T} u(w)}{|\{w : w \vDash T\}|}$
- Optimistic voters: $ut^o(X) = \max\ \{u(w) : w \in X\}$
- Pessimistic voters: $ut^p(T) = \min\ \{u(w) : w \in X\}$
- Expected value voters: $ut^e(X) = \sum_{w \in X} u(w)p(w)$

  Above, if we assume the probability p to be uniform, then $p(w)$ will be just $\dfrac{1}{|x|}$.

Note that with ut^e we will have a convexity property. If X,Y are disjoint, then $ut^e(X \cup Y)$ will be in the closed interval whose endpoints are $ut^e(X)$ and $ut^e(Y)$.

15.7.2 The Value of a Message

- Suppose T is the logical closure C of T_c.
- What's the best thing for her to say next?
- Roughly: $val(A, T) = ut(T \circ A) - ut(T)$.
- $T \circ A$ is what T becomes after A is added,
- And $val(A, T)$ is the value of uttering A when her current theory (as seen by voters) is T.

  But the precise definition will depend on

- The kind of voter we're assuming (i.e. o vs. p vs. e)
- The set from which A is selected.

  With respect to the latter, consider A from

- $\chi_a = T_a$ (i.e., only "true convictions")
- $\chi_t = \mathrm{L} - \{\neg A : T_a \vdash A\}$ (i.e., anything consistent with "true convictions" = **tactical**)
- $\chi_m = \mathrm{L} - \{\neg A : T_c \vdash A\}$ (i.e. anything consistent with the current theory = **Machiavellian**)
- $\chi_l = \mathrm{L}$ (i.e., any sentence in the language, allowing for contradictions and lying)

Note: $\chi_a \subseteq \chi_t \subseteq \chi_m \subseteq \chi_l$.
If we have $\chi = \chi_l$ then T_c may become **inconsistent**.
In this case, $\circ = *$ (i.e. an AGM-like update operation).
In the other cases $\circ = \dot{+}$ addition of A followed by logical closure.
If $\chi = \chi_a$, χ_t or χ_m then we let

$$val(A,T) = ut(T \dot{+} A) - ut(T)$$

where ut is one of ut^o , ut^p or ut^e.
We can now define **best statements** for C given T from χ as follows:

$$best(T,\chi) = argmax_A\, val(A,T) : A \in \chi.$$

15.7.3 Complex Statements

Proposition 1 (1). Assume e-voters. For all A, B such that A , B , $A \wedge B \in \chi_m$ (i . e., A, B, $A \wedge B$ consistent with T_c) there exist a , . . . $f \in [0, 1]$ such that

1. $a \cdot val(A, t) + b \cdot val(\neg A, T) = 0$
2. $val(A \wedge B,T) = val(A,T + val(B,T \dot{+} A)$
   $= val(B,T) + val(A,T \dot{+} A)$
3. $c.val(A \vee B,T) + d \cdot val(A \wedge B,T) =$
   $e \cdot val(A,T) + f \cdot val(B,T)$

<u>Proof:</u>
For 1):

$$ut(T) = a \cdot ut(T + A) + (1 - a) \cdot ut(T + \neg A)$$

where

$$a = \frac{\{w | w \models T \dot{+} A\}}{\{w | w \models T\}}.$$

15.7.4 Moving to Complete Theories

Corollary 1 *There is a* ***complete*** $T \subseteq T_c$ such that $ut^e(T) \geq ut^e(T_c)$**.**
<u>Proof</u>: From the above, we must have exactly one of

(i) $val(P_i, T) = val(\neg P_i, T) = 0$
(ii) $val(P_i, T) > 0$ and $val(\neg P_i, T) < 0$

(iii) $val(P_i, T) < 0$ and $val(\neg P_i, T) > 0$

Suppose $Q_i, \ldots, Q_k (k \leq n)$ are all the atoms not in T_c.

Let $T_0 = T_c$ and $T_{i+1} = \begin{cases} T_i \cup Q_i & val(Q_i, T_i) \geq 0 \\ T_i \cup \neg Q_i & else \end{cases}$

Let $T = Cn(T_k)$

Corollary 2 *One of the best extensions of T_c is a complete theory $T \subseteq T_c$.*
Proof.

- Suppose T' is a best extension of T_c and T' is incomplete.
- By the previous corollary, there is $T'' \supseteq T'$ which is a complete extension of T' (and thus of T_c) such that $ut^e(T'') \geq ut^e(T')$.
- T'' is complete and among the best extensions.
- The previous result suggests that if C assumes e-voters, then it will never be to C's disadvantage to move towards a complete theory.
- This will also be the case if the voters are either e-voters or p-voters.
- But why then do we have *The Onion* phenomenon?
- I.e., Why do candidates state vacuities like "`God bless America`" or "`9/11 was a tragedy`"?
- *Conjecture*: They must be assuming that there are at least some o-voters (who 'always assume the best').

$$T \supseteq T' \Rightarrow \max\{u(w) | w \vDash T'\} \leq \max\{u(w) | w \vDash T\}$$

I.e., $T \supseteq T' \Rightarrow ut^o(T') \leq ut^o(T)$

15.8 Conclusions

When we are programming people, the task is much more complex than it is with computers.

- People have their own motivations.
- Information which people need in order to act properly must be made available to them and sometimes, some information may need to be hidden.
- When different people have opposing motives, then conflicts can arise.

Nonetheless, issues arise in *Social Software* which are similar to the issues which arise in programming.

- We need to be clear about the desired post conditions of the procedure we propose.
- We need to make sure what are the preconditions needed for the procedure to work.

- Exchange of information and preserving the proper order of actions must be attended to just as it is in Distributed Computing.

Some of my recent papers can be downloaded from the site http://cuny.academia.edu/RohitParikh.

At the time I wrote this paper I was not sufficiently familiar with Searle's book *Making the Social World: The Structure of Human Civilization* (2010), and had forgotten the relevance of Hohfeld's *Fundamental Legal Conceptions as Applied in Judicial Reasoning* (1923/1978). But consider. If Jane wants to cross the river she may use the bridge. If there is no bridge, she may build one. But before she does that, she needs a permit to build the bridge. Both Hohfeld and Searle point to this important variety of social algorithms which will surely be the subject of a future paper. For information about animal minds, (Lurz, 2011) is a good reference. For further details on this research project see (Parikh 2002, 2009).

Acknowledgment The research reported in this paper was supported in part by a grant from the PSC-CUNY FRAP program.

References

Aumann, R. 1976. Agreeing to Disagree. *Annals of Statistics* 4: 1236–1239.

Brams, S. 2007. *The Presidential Election Game*. New Haven: Yale University Press.

Chwe, M. 2001. *Rational Ritual*. Princeton: Princeton University Press.

Dean, W., and Parikh, R. 2011. The Logic of Campaigning. In *Presented at Logic and Its Applications, Lecture Notes in Computer Science, Volume 6521, 4th Indian Conference, ICLA* 2011, Delhi, India, January 5–11, 2011.

Hayek, F. 1945. The Use of Knowledge in Society. *The American Economic Review* XXXV (4): 519–530.

Hohfeld, W. N. 1923/1978. *Fundamental Legal Conceptions as Applied in Judicial Reasoning: And Other Legal Essays*. New Haven: Yale University Press. Reprinted from the 1964 edition as a paperback.

Kamien, M.I., Y. Tauman, and S. Zamir. 1990. On the Value of Information in a Strategic Conflict. *Games and Economic Behavior* 2: 129–153.

Lewis, D.K. 1969. *Convention: A Philosophical Study*. Cambridge, MA: Harvard University Press.

Lurz, R. 2011. *Mindreading Animals: The Debate Over What Animals Know About Other Minds*. Cambridge, MA: MIT Press.

Neyman, A. 1991. The Positive Value of Information. *Games and Economic Behavior* 3: 350–355.

Pacuit, E., and R. Parikh. 2006. Social Interaction, Knowledge, and Social Software. In *Interactive Computation: The New Paradigm*, ed. D. Goldin, S. Smolka, and P. Wegner. New York: Springer.

Parikh, R. 2002. Social Software. *Synthese* 132 (3): 187–211.

———. 2009. Knowledge and Structure in Social Algorithms. In *Relations and Kleene Algebra in Computer Science*, ed. J. Berghammer and B. Möller, 1–12. New York: Springer.

Parikh, R., Ç. Taşdemír, and A. Witzel. 2013. The Power of Knowledge in Games. *International Game Theory Review* 15 (4): 1–28.

Premack, D., and G. Woodruff. 1978. Does the Chimpanzee Have a Theory of Mind? *Behavioral and Brain Sciences* 1 (4): 515–526.

Searle, J.R. 2010. *Making the Social World: The Structure of Human Civilization*. New York: Oxford University Press.

Sen, A. 1999. *Commodities and Capabilities*. New York: Oxford University Press.

Stalnaker, R. 2005. Saying and Meaning, Cheap Talk and Credibility. In *Game Theory and Pragmatics*, ed. A. Benz, G. Jaeger, and R.V. Rooij. Basingstoke: Palgrave MacMillan.

Vickrey, W. 1961. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance* 16 (1): 8–37.

Index

A
Algorithm, 2, 8, 17, 22, 27, 49, 54–59, 70, 104, 110, 112, 153, 157–158, 190, 212, 234, 255, 268, 324, 339–356
Artificial intelligence (AI), 2, 11, 18, 21–25, 30, 103, 156–157, 220, 238, 244, 245, 265, 267–269, 276, 302, 306, 308, 331
A-3 scrambler, 18, 160–162, 164, 165, 174, 175, 186
Automatic Computing Engine (ACE), 12, 154, 155
Axiom of infinity, 83, 84

B
Babbage, C., 40, 155, 243, 267
Baby, 10, 12, 190–193, 206, 210
Bell Labs, 19, 166, 174, 183, 189
Bletchley Park, 6, 11, 30, 133, 142, 182, 192, 199, 250
Brain, 18, 21, 24, 26, 156, 197, 235, 237–246, 251, 259, 268–270, 276, 277, 298, 301, 306, 307, 309–313, 315–318, 325, 326, 328–330, 341

C
λ-Calculus, 8–10, 69, 70, 72, 75
Carnap, R, 111–113, 121, 127, 128, 141, 143
Causality, 17, 21, 30, 234–238, 240, 241, 248, 249, 255, 257, 310
Cells, 192, 220, 241, 288, 289
Chalmers, D., 298–301
Church, A., 8–11, 27, 29, 40, 42, 44, 45, 47, 58–59, 66, 67, 69–72, 75–77, 80, 106, 113, 126, 219, 250, 256
Church's Thesis, 9, 27, 29, 44–46, 64, 69, 70, 77, 79
Church-Turing thesis, 9, 234, 249, 253, 256, 339–356
Ciphony, 18, 19, 159, 162, 164, 182, 186
Circle-free, 130–132
Circular machine, 130
Circularity, 15, 45, 65, 68, 71–72, 76, 78, 87, 130–131
Classical computational theory of mind (CTM), 103–105, 142, 223, 237, 240, 279–281, 283, 287, 294, 297, 302
Colossus, 156, 192
Configuration, 24, 42, 46–48, 50–52, 56, 110, 130, 220, 288
Consciousness, 3, 237–239, 241, 266, 311, 316
Cultural search, 325, 328

D
Decidability, 14, 41, 42, 45, 52, 54, 58, 60, 68, 104, 108, 118, 127
Decision problem, 6, 43, 59, 112, 343
Definability, 8, 10, 11, 14, 15, 21, 22, 44, 64, 68, 70, 78, 82–87, 108, 111, 128, 129, 254–259
Delilah, 19, 159, 175, 182, 183, 185, 186, 244

© Springer International Publishing AG 2017

J. Floyd, A. Bokulich (eds.), *Philosophical Explorations of the Legacy of Alan Turing*, Boston Studies in the Philosophy and History of Science 324,
DOI 10.1007/978-3-319-53280-6

E
EDVAC, 9, 154, 155
Einstein, A., 4, 230, 236, 241
Emergence, 13, 21, 104, 235, 245–249, 253, 256–257
Emotional concept, 305–319
Enigma, 11, 133, 184
Entscheidungsproblem, 6–8, 14, 39–42, 52, 54, 59, 60, 77, 105, 107, 109, 110, 118, 120, 124, 125, 128, 131, 219, 234, 250

F
Fodor, J., 24, 279–283, 285–288, 291, 293, 296–298, 302
Formalism, 14, 15, 27, 45, 47, 63, 66–69, 71, 73–75, 78, 81–85, 107, 108, 119, 121, 127, 136, 141, 143, 156, 219, 224, 279–281, 288
Formal-syntactic conception of computation (FSC), 24, 25, 279–281, 286, 287, 290, 292, 293, 296, 302
Formal syntactic item, 279–281, 291–293
Formal syntactic properties, 281, 282, 287, 290, 292, 301, 302
Free will, 256, 305–319

G
Genesis, 23, 265, 271–276
Genetic evolutionary search, 325, 326, 328
Gettier, E.L., 15, 93–101
Gödel, K., 6–10, 14, 15, 22, 39–45, 52–54, 58–59, 63, 78–88, 106–108, 110, 112, 113, 128, 132, 134, 156, 234, 243, 248

H
Halting argument, 41, 43, 56, 57, 59, 88, 132, 250, 253
Hardy, G.H, 16, 40, 108, 114, 117–122, 126, 134–136, 138, 139, 142
Hilbert, D., 7, 8, 14, 22, 39–45, 47, 52, 59, 64, 68, 69, 71–73, 107, 110, 112, 119, 126, 127, 129, 135, 219, 250

I
Imitation game, viii, 265, 305, 307, 315, 316, 318, 319 (*see also* Turing test)
Incompleteness Theorem, 40, 53, 58, 64, 72–77, 82, 107, 108, 125, 126, 128, 132, 134, 142, 156, 243, 250, 256
Indeterminism, 313, 318
Intellectual search, 27, 325, 326, 328, 329
Intelligence, 2, 13, 17, 18, 21–26, 29, 103, 154, 156, 157, 161, 162, 265–277, 306–308, 314, 315, 317–319, 326, 328–331
Intelligent machinery, 12, 190, 305, 306, 323, 324, 327–329, 331

J
Justification, 2, 15, 16, 94–98, 100, 132, 315

K
Kilburn, T., 190–192, 201
King's College, 5, 29, 112, 118, 123, 124, 195
Kleene, S.C., 8, 9, 44, 66, 69, 70, 72, 75, 77–79, 81, 86, 106, 113, 255

L
Language–game, 109, 117, 118, 132, 135
Lovelace, L., 266, 267, 269, 314–316

M
Machine intelligence, 328, 331
Machine learning, 27, 269, 324
Mapping, 20, 63, 221–222, 335
Mark II, 193, 194, 213, 215, 216
Measurement based quantum computers (MBQCs), 221, 224, 226–231
Mentalese, 24, 281–287, 289–291, 295, 297

N
Narrow content, 25, 285
Never Ending Language Learning (NELL), 26, 331–333, 336
Newman, M., 3, 4, 6, 8, 11, 12, 39, 105, 112–115, 121, 124, 195–198, 241, 308

O
Oracle, 10, 21, 125, 138–140, 257, 258
Ordinal logic, 10, 125, 129, 143, 249

P
Piccinini, G., 20, 222–224, 230, 232
Poincaré, H., 67, 245, 246, 256
Polynomial time, 14, 53–58

Principia Mathematica, 12, 27, 40, 72, 73, 121, 123, 128, 129, 140, 325
Provability, 14, 15, 64, 68, 70, 74, 76, 82–83, 85, 87
Putnam, H., 20, 24, 102, 125, 129, 222, 244, 279, 282, 285–287, 300

Q
Qubits, 20, 224–230

R
Recursive function, 9, 44, 45, 53, 69, 75, 77, 81, 112, 129, 253, 258
Representational theory of mind (RTM), 280–284, 290, 301
Response-dependent, 25, 307, 316, 318
Rules, 7–9, 20, 23, 28, 33, 39, 44, 45, 47, 48, 67, 68, 71, 73, 75–78, 111, 116, 117, 121, 127, 130, 132, 134, 137, 219, 223, 225, 229, 230, 251, 271, 272, 274, 276, 281, 287–290, 295, 301, 302, 307, 308, 319, 324, 325, 331, 333, 345
Russell, B., 4, 10, 17, 39, 40, 108, 109, 113–115, 117–119, 121–124, 126, 128, 129, 133, 137, 141, 248, 284, 325

S
Scan, Scanner, 47, 50, 53, 77, 111, 183, 288, 289, 306
Scope problem, 72–78, 87
Searle, J., 20, 24, 156, 222, 279, 356
Semantically permeated (SP), 25, 281–294, 301, 302
SIGSALY, 18, 47, 49, 52
Simulate, 40, 46, 49, 130, 166, 220, 221, 224, 227, 228, 231, 235, 237–241, 245, 268, 306
Software, 12, 17–19, 28, 50, 104, 155, 157, 192, 210, 211, 216, 333, 339, 355
State of mind, 24, 25, 51, 95, 109, 135, 136, 306, 307, 315
Strachey, C., 19, 195–197, 214
Subroutines, 19, 105, 199, 200, 202

T
Tarski, A., 75, 81, 111–113, 128, 129, 254
Tautology, 56, 57, 59, 114, 123, 130, 132
Turing machine, 6, 8, 10, 15–17, 20, 21, 40, 41, 43, 46–59, 64, 65, 77–81, 87, 88, 95, 108, 112, 130, 136, 142, 155, 190, 192, 219–224, 231, 234, 241, 243–245, 250, 253, 257, 279, 281, 288–290, 295, 306
Turing test, 12, 26, 103, 265, 277, 308

U
Universal machine, 2, 8, 11, 17, 18, 26, 29, 40, 49, 51–53, 60, 104, 131, 153, 154, 156, 190, 220, 243, 244, 246, 325

V
Vienna Circle, 113, 114
Vocoder, 18, 19, 165–172, 176, 182, 183, 186
von Neumann, J., 5, 6, 9, 14, 40, 54, 58, 59, 73, 154, 155, 269

W
Watson, A., 124–125, 130, 134
Watson (IBM's), 266, 331, 333–337
Wide content, 25, 285, 289, 294, 295
Williams, F., 190–192, 200, 201
Wittgenstein, L., 5, 16, 18, 22, 28, 66, 93, 96–97, 100, 104, 106–139, 141, 142, 253